ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2000/00(00)-0000-00
doi: 10.3778/j.issn.1673-9418.

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-89056056

DevOps 过程能力支持框架模型及其实现

顾胜晖，张贺，邵栋+

南京大学 软件学院，江苏省 南京市 210093

A Process Capability Support Model for DevOps and Its Implement

GU Shenghui, ZHANG He, SHAO Dong+

Software Institute, Nanjing University, Nanjing 210093, China

+ Corresponding author: E-mail: dongshao@nju.edu.cn

Author. A Process Capability Support Model for DevOps and Its Implement. Journal of Frontiers of Computer Science and Technology, 2000, 0(0): 1-000.

Abstract: As an emerging approach to support fast delivery of software features with reliable quality, DevOps attracts more and more practitioners and shows the potential to become one of the mainstream approach for software development and operation. Many universities begin to offer DevOps related courses to the students majored in software engineering and computer science. However, as a critical part of a DevOps course, the project practicing using DevOps might cast big challenges for teachers, compared to traditional project practicing. It becomes teaching obstacles to offer practicing environment, process management and evaluation approaches. This paper introduces a model—*DOPCSM* to support the management and evaluation of student teams practicing DevOps. By integrating several popular open source tools, teachers could implement the model into a support system, which provides students with features such as group management, project management and student performance data analysis, etc. Meanwhile, the support system also provides teachers with sufficient evidence to perform evaluation. Our preliminary trial in Nanjing University revealed several advantages of the *DOPCSM* model.

Key words: DevOps, Process Capability Support Model, Project Management, Software Engineering

摘 要：作为一种支持快速交付同时确保软件质量的新兴方式，DevOps 吸引了愈来愈多的实践者，并且有望成为未来软件开发以及运维的主要模式。如今许多高校开始向软件工程及计算机科学专业的学生提供 DevOps 相关的课程。然而，作为 DevOps 课程的关键部分，与传统项目实践相比，DevOps 项目实践可能会给教师和学生带来巨大的挑战。项目的实践环境、过程管理方式以及评估手段成为了教学的难点。由此论文提出了一个支持学生团队 DevOps 实践管理与评估的模型——*DOPCSM*。教师可以通过整合不同的开源 DevOps 工具来实现此模型，开发后的支持系统可以向学生提供团队、项目管理等特性，与此同时，还向教师提供充足的项目评估证据。*DOPCSM* 模型在南京大学开展的试验中展现出了的积极的作用。

关键字：DevOps；过程能力支持模型；项目管理；软件工程
文献标志码：A 中图分类号：****

1 引言

近年来，DevOps 作为一种新兴的文化与哲学，以一种持续的形式演化软件产品，并且将软件开发生命周期的各个部分进行无缝整合。DevOps 的产生是为了协调开发和运维团队，缓解他们之间的矛盾冲突，以此达到提高产品交付速率，同时保障产品质量的目标。当我们提及 DevOps 文化时，自动化是一个重要的概念。高程度的自动化是短周期持续交付的基础[1]，另外也是获得快速反馈的关键[2]。

在实践过程中，实际体现 DevOps 自动化的是各种类型的工具，尤其是由 DevOps 工具整合而成的流水线(Pipeline)。现如今，在一些知名企业的带领下，比如 Facebook、Yahoo、Netflix 和 Flickr，DevOps 文化已经逐步被愈来愈多的软件组织所接纳[3-6]。正是由于 DevOps 文化的普及，许多软件企业接连开发出持续交付(Continuous DElivery, CDE)或持续部署(Continuous Deployment, CD)的流水线，比如 IBM、Amazon 以及 Microsoft[7-9]。这些流水线和工具能够支持和促进持续交付，加速问题的解决速度，最终在快速、不断的需求变更的压力下满足用户的要求。

鉴于 DevOps 的优势，将 DevOps 文化引入课堂成为了一种需求。通过课堂中的 DevOps 教学，学生们可以体验和理解实际工业环境中的 DevOps 实践，感受 DevOps 的特性，为他们今后的职业生涯打下基础。虽然当前有许多在线的课程[10-11]，但是它们的主题大多是对现有工具的使用方法的介绍，强调的是企业级别的实际应用，而不是帮助学生体验 DevOps 过程方法的优势。在学术界，已经有部分研究员提出了一些 DevOps 教学方案，但遗憾的是，这些方案并没有关注 DevOps 自动化实践的过程[12-13]。

目前有许多问题和挑战阻碍着 DevOps 教育的开展，其中最主要的是两类挑战：技能需求和技术环境[14]。学生需要掌握运维技巧并需要学会配置开发运维环境。对教师而言，他们需要有恰当的方法去评估学生的表现。除此之外，在使用众多复杂 DevOps 工具的情况下，学生之间的协作也会变得问题重重。基于以上原因，一个提供实践环境并支持自动分析的工具就变得尤为重要。

在这篇文章中，我们提出了一个过程能力支持模型—*DOPCSM(DevOps Process Capability Support Model)*。该模型抽象并定义了 DevOps 项目实践中的具体过程以及必要的特性。基于该模型，教师可以实现适合自己教学需求的支持系统，通过支持系统帮助学生更便捷、轻松的完成 DevOps 项目实践，同时实现学生项目的监控和自动评估。

本文剩余部分结构如下：第二部分概述了研究的相关工作；第三部分介绍了模型的设计以及特性；第四部分详细阐述了项目评估指标的设计；在第五部分，我们实现了模型的原型系统，并详细说明了模型在实践中的应用；第六部分讨论了模型的潜在问题；我们在最后一部分进行了总结并提出了未来的工作方向。

2 相关工作

2.1 软件工程教学中的工具

许多实践者和学者认为，项目实践是软件工程教学的重要一环。项目实践的主要目标是通过一个真实的或近似真实的软件项目环境来帮助学生将理论应用于实践。通常来说，项目实践分成两个类别：技术指向性项目实践(Technology-specific project practicing)[15-16]，以及过程指向性项目实践(Process-specific project practicing)[17-20]。现在有许多工具支持这两个类别项目实践的评估。

Raza 等人提出了一个支持开发者表现评估的工具—*ProcessPAIR*[21]。此工具可以自动检测潜在的问题，并对它们定级，同时通过从许多开发者表现数据分析得到的性能模型来定位问题的根源。除了 *ProcessPAIR*，还存在许多支持过程教学的工具，比如 PSP(Personal Software Process)[22-23]和 TSP(Team Software Process)[24]。

2.2 DevOps 工具

随着 DevOps 文化的深入人心，越来越多的 DevOps 工具如雨后春笋般涌现。这些工具种类丰富、功能繁多，包括持续集成工具、配置管理工具、容器等等。另外，版本控制系统和项目管理系统在某种程度上也可以被列为 DevOps 工具。

然而，DevOps 持续上升的热度导致自动化工具的数量爆炸。根据 XebiaLabs[1]（一个致力于探索 IT 行业前沿技术并提供解决方案的知名组织）的研究显示，目前大约有 15 个种类，150 种 DevOps 相关工具。在持续集成的类别中有 12 个工具，包括 *Jenkins*、*Bamboo*[2]、*Travis CI*[3]等；有 14 个工具被列为部署工具，比如 *Juju*[4] 和 *AWS CodeDeploy*[5]等。

面对数量如此庞大，种类如此繁多的 DevOps 工具，教师甚至是一些专家也苦于学习工具的使用方法，更别提学生。另外，他们还需要花费时间和精力去设计工具间该以何种方式协作，以便完成一个特定的项目。在教学情况下，无论对教师还是学生而言，这都是一件费时又费力的工作。因此，我们希望设计一个模型来定义 DevOps 工具的使用流程，并且提供一种便捷的管理方式。

2.3 DevOps 教学中的挑战

在软件工程实践中，DevOps 逐步成为了一种趋势，而且在未来的软件开发和运维中，DevOps 可能会成为主导的方式。然而，现在大多数传统的软件工程课程仅仅只关注软件生命周期的前中期（也就是需求、设计、编码、测试和组装阶段），而产品阶段往往被忽视（也就是部署和维护阶段）。简而言之，他们只是在教授“Dev”而非“Ops”[14]。

因此，越来越多的教师致力于将 DevOps 带入课堂，但是却遇到了许多挑战。Christensen 识别了 DevOps 教学中的 5 种挑战：a) 教师的经验；b) 技能的种类；c) 技能与实践的结合；d) 实践环境，以及 e) 评估和计分机制[14]。另外，Christensen 将这些挑战（除了教师的经验）分成了两个种类：技能需求和技术环境。具体来说，技能需求指的是 DevOps 教学更需要的是学生的实践技能，而非书本知识；技术环境意味着教学过程中需要一个强大的平台环境来支持学生的实践，同时需要有一个有效的学生表现评估机制。

为了解决上述的挑战，有些学者已经提出了一些定制的教学方法。接下来的段落将简要介绍他们的工作。

Ohtsuki 等人提出了一个整合了不同 DevOps 工具的教学支持系统—*ALECSS*，用于提升软件质量并向学生和教师提供快速反馈[25]。*ALECSS* 使用 *Git* 作为源码管理工具；*Jenkins* 作为持续集成工具；*Ant* 作为构建工具；*FindBugs*[6]、*Checkstyle*[7]和 *JUnit* 作为测试工具。该系统的主要功能是自动向学生和教师提供评估报告。此外，他们还通过比较 DevOps 工具以及 *ALECSS* 生成的报告来对系统做初步的评估。然而，*ALECSS* 系统只是通过缺陷检测来解决 DevOps 教学评估产生的挑战。关注的重点是项目代码的质量而不是 DevOps 实践的整体过程，学生无法从系统中感受 DevOps 带来的优势，评估的侧重点也只是在代码质量而非过程质量。

Eddy 等人提出了 *CDEP*(Continuous Delivery Educational Pipeline)来解决软件工程课程中持续集成和持续交付教学引入的问题[26]。与 *ALECSS* 类似，*CDEP* 通过集成各种 DevOps 工具来实现流水线。它的主要目标是提供一个便携的、可重用的、可视的以及可快速配置的教学工具。如果教师对复杂的工具不熟悉，*CDEP* 能够帮助他们关注在更重要的事情上而不是整个流水线繁琐的配置。事实上，*CDEP* 只是通过流水线解决了实践环境的问题。

虽然已经有一些支持系统存在，但是仍然需要一种能够同时应对两类挑战（也就是技能需求和技术环境）的解决方案。当提及 DevOps 教学工具时，应该有一个全面的、易用的、标准化的工具或系统来提供开箱即用

[1] http://tinyurl.com/p3emhuz
[2] http://tinyurl.com/mtlkxuu
[3] http://tinyurl.com/y8og8jlv
[4] http://tinyurl.com/ngs5ec7
[5] http://tinyurl.com/lw3sopl
[6] http://tinyurl.com/346lac
[7] http://tinyurl.com/8k4w

的 DevOps 实践环境以及自动评估手段。

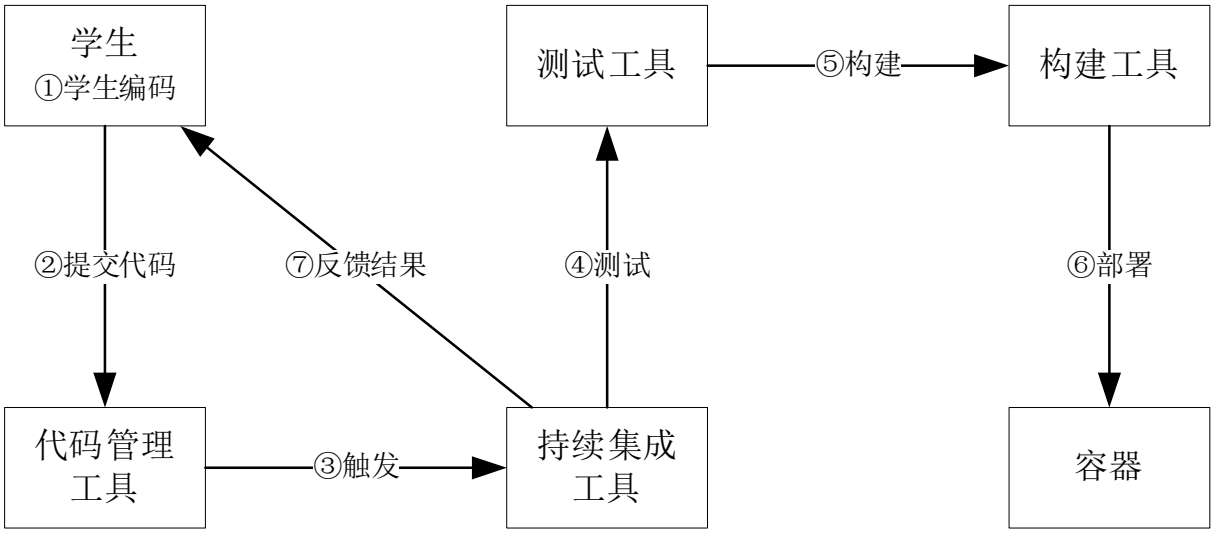

Fig. 1 Workflow of DevOps practice

图 1 DevOps 实践的工作流

3 模型设计

3.1 概述

为了弥补现有 DevOps 教学过程中的不足，本文提出了一个过程能力支持模型——*DOPCSM*。此模型定义了现有的 DevOps 开源工具的整合方法，并且提供一些以教学为目的的特性，学生的 DevOps 实践过程流程可以依据此模型开展。*DOPCSM* 模型旨在使学生更关注开发的过程以及代码本身，而不是繁琐的工具配置和使用。另外，*DOPCSM* 定义了学生团体实践过程管理的手段，提供了快速持续反馈的解决方案，学生可以通过这些反馈持续改进代码质量。对教师来说，他们可以根据反馈来评估项目的过程质量以及每个学生的表现。*DOPCSM* 模型能够自动收集必要的信息，为教师在学期末评估整个学生项目提供数据支持。本模型减轻了教师的手工评估的工作量，同时也提供了更简洁的 DevOps 实践环境。

在实现 *DOPCSM* 模型的过程中，教师可以整合不同种类的 DevOps 工具。在源代码管理和版本控制阶段，可以使用 *Git/GitHub*、*GitLab*、*Subversion*、*TFS(Team Foundation Server) Version Control*[8]等工具[12]；持续集成或部署服务器可以选用 *Jenkins*、*TeamCity*[9]、*Travis CI* 等；在测试阶段，可以使用像 *SonarQube*[10]一样的持续检测平台，也可以分别使用像 *Junit*、*FindBugs* 和 *Checkstyle* 一样的特定工具；在构建阶段，可以根据项目选用 *Gradle*、*Maven*、*MsBuild*[11]等构建工具；最后在生成制品后，可以使用 Docker 容器作为部署工具。

在接下来的几章节中，本文将详细介绍 *DOPCSM* 模型的设计以及特性。

3.2 DevOps 实践工作流

DOPCSM 是一个过程能力支持模型，它提供了个人、团队和项目的同步管理手段，定义了数据收集和分析的方法，并且为学生和教师提供了分析结果反馈的功能。所有的过程状态数据都是从项目开发过程中获取的。然而，不同的项目团队可能使用不同的工作流(Workflow)以及不同的 DevOps 工具。因此，为了使教学过程可控，需要预先定义好一个工作流，否则教师会发现持续监控和评估多个学生项目会变得非常困难。作为 *DOPCSM* 模型的前置条件，本文定义的工作流是一个循环流水线，旨在帮助开发者持续提升代码质量。在接下来的段落中，所有的步骤将会被详细说明，整体的工作流如图 1 所示。

步骤 1：学生在自己本地的集成开发环境或文本编辑器中编写源码。

[8] http://tinyurl.com/y9xcc59g
[9] http://tinyurl.com/lerbqd6
[10] http://tinyurl.com/y7lsunpp
[11] http://tinyurl.com/mcgmw33

步骤 2：在完成特定阶段的编码后，比如完成了一个新功能或修改了一个缺陷，学生可以将代码通过版本控制系统提交到代码仓库中。

在这个阶段，开发者可以选用自己喜爱的版本控制系统和代码仓库，其中比较知名的是 *Git* 和 *GitHub*。*Git* 是一个开源的版本控制系统，它的优势在于分布式存储；*GitHub* 是 *Git* 的云代码仓库，提供免费的存储空间，且支持不同开发者的协作开发。在模型实现过程中，还可以使用其他的版本控制系统，只要它与持续集成系统相兼容。

步骤 3：一旦源码被更新了，持续集成服务器会自动触发构建任务。

现在越来越多的工具开始提供流水线功能，其中最典型的就是 *Jenkins*。*Jenkins* 是开源自动化服务器的领导者，旨在支持自动化构建和制品的自动部署，使持续集成甚至持续部署变得可能[1][27-28]。自 *Jenkins 2.0* 以来，代码流水线(Code pipeline)就成为了其重要的特性之一。通过编写流水线脚本（在 *Jenkins* 中是 *Jenkinsfile*），开发者几乎可以执行任何任务。其它持续集成工具也提供类似的功能。在本模型中，代码流水线通常包括以下几个阶段：1) 从代码仓库中获取源码；2) 通过测试工具执行测试；3) 通过构建工具构建源码并生成制品；4) 将通过测试的制品部署到服务器中。

步骤 4：通过测试工具执行一系列测试，最后生成报告。

测试阶段包括静态测试、单元测试、覆盖率测试、集成测试等，具体测试种类根据测试工具和实际环境的不同而调整。开发人员可以分别使用特定功能的工具完成测试，比如静态测试使用 *FindBugs*，集成测试使用 *JUnit*，覆盖率测试使用 *Cobertura*[12]等；还可以使用测试平台，其中较为典型的是 *SonarQube*。*SonarQube* 是一个开源的源码质量管理平台，能够检测项目应用的健康程度以及发现新引入的问题。它最大的优势是通过插件的形式集成了不同种类的测试工具。而且 *SonarQube* 不仅仅只是简单的将各个工具的检测报告糅合在一起，它通过独特的算法将这些报告重新处理，最后量化的度量整体质量。基于本模型的支持系统应当整合测试报告，并根据需求选择合适的数据，以便针对性的帮助开发者提升代码质量和评估项目。

步骤 5：测试完成后，构建工具会编译源码并生成最终制品。

本模型的支持系统应当根据项目使用构建工具的变化而变化。最终的构建信息会被收集和分析。

步骤 6：如果代码测试通过了，且构建成功，则最终制品会被部署到容器中并且自动开始运行。

与虚拟化技术相比，容器技术在持续部署环境下会有更好的性能。其中 *Docker* 是一个比较出名的开源容器引擎[29-30]。因此本模型定义使用容器技术作为部署环境。

步骤 7：学生根据反馈报告修改项目代码，并且从步骤 1 开始下一个循环。

整个工作流是为学生体验 DevOps 开发优势而设计的。在面对教师提出快速变更的需求的情况下，传统的开发方式可能会导致产品交付的延期，并可能因为未修复问题的大量积压而最终提交一个糟糕的产品。相较于传统开发模式，DevOps 开发方式可以弥补上述缺陷，在确保产品质量的同时满足持续部署的开发需求。

3.3 DOPCSM 模型

基于 3.2 节的工作流，本文提出了一个过程能力支持模型—*DOPCSM*。*DOPCSM* 定义了实践过程中使用的 DevOps 工具的协作方式，定义了学生使用过程中的管理方法，以及数据的获取与分析手段。学生可以仅使用基于模型实现的系统，而不是直接管理多个 DevOps 工具，减轻了学生使用的成本。教师可以直观的获得评估报告，而不是手动去各个工具中寻找数据然后评估，减轻了教师评估的成本。模型的整体结构和特性如图 2 所示。

[12] http://tinyurl.com/lhpstsk

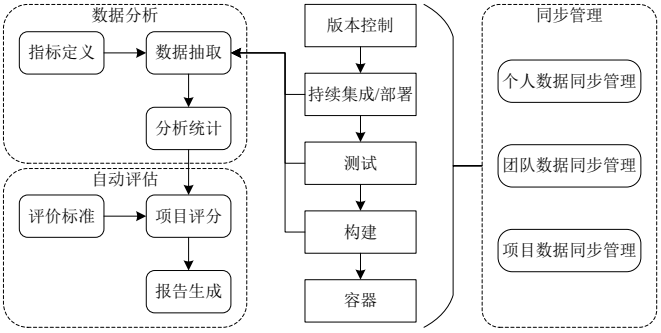

Fig. 2 DOPCSM model

图 2 DOPCSM 模型

3.3.1 个人数据同步管理

对学生来说，管理各个 DevOps 工具中个人的信息是繁琐和不现实的。举个例子，学生需要在 *Jenkins* 和 *SonarQube* 中分别注册账户，然后需要更新个人信息，管理团队之间开发者的关系。另外，各个工具数据的不通用也给教师的评估带来了困难，教师需要清楚的知道每个学生在不同工具系统中的账号，这就增加了管理的成本。

DOPCSM 模型定义了个人数据同步管理的方法，即通过 API 在各个整合的 DevOps 工具系统中同步创建和更新数据。使用这种方法，最终用户只需要提供一次信息就能够在多个系统中同步数据，比如注册的时候用户只要提供必要的个人信息，支持系统就会向各个 DevOps 工具同步注册。目前大多数 DevOps 会提供丰富的 API，其中包括用户信息的管理。

3.3.2 团队数据同步管理

在教学过程中，组队完成一项任务是非常常见的，可是同时管理多个工具的团队数据（比如团队成员、团队成员权限）也是极其繁琐的，有些工具系统（比如 *Jenkins*）甚至不提供团队功能，这就给学生团队实践带来了阻碍。另外，如果没有团队数据管理，学生在团队中的贡献也会难以度量。在教学情境下，团队管理是一个必要的功能。

DOPCSM 模型定义了团队数据同步管理的方法，即团队管理功能在本地实现，并通过 API 同步到支持团队管理的 DevOps 工具中。团队管理功能在本地实现可以解决 DevOps 工具不支持团队管理的问题。团队成员和成员权限等信息会存储在本地，如果可能，则通过 API 同步到 DevOps 工具中。

3.3.3 项目数据同步管理

与个人数据类似，项目数据的分散管理也是一个繁重的任务。如果没有同步的方法，删除项目的操作就需要在不同工具中重复执行。此外，教学的实践项目可能会有特殊要求，比如项目需要是一个基于 Web 的系统。在这种情况下，项目的初始化对学生来说还是有一定难度的，失败的配置经历可能打消学生的积极性。

DOPCSM 模型定义了项目数据同步管理的方法，即提供项目模板并自动初始化项目，同时基于 API 提供管理功能。学生只需要提供项目基础信息，比如项目名称，就能够一次性创建和初始化特定的项目类型。支持系统应当通过 API 使用预先定义的模板生成项目配置。在实践过程中，学生可以通过支持系统同步更新或删除项目的信息。

3.3.4 数据获取与分析

虽然 DevOps 工具会提供各种各样的数据，但是并非所有的数据都是适合用于评估的。此外，有些数据是无法获取或者没有提供的。

为了解决这些问题，*DOPCSM* 模型定义数据获取和分析的方法，即通过 API 或者爬虫抓取需要的信息，并通过适当的算法对数据进行分析和统计，以便生成自动评估报告。在这过程中，用户首先需要定义指标，支持系统根据指标抽取合适的数据，然后进行统计分析。举例来说，用户想要从 *Jenkins* 中获得构建的频率，可是 *Jenkins* 中没有现成的数据。于是支持系统会抽取所有构建的时间点，根据时间点和构建次数计算出构建的频率。

3.3.5 自动评估机制

现有的系统大多数不会提供自动评估的方法，教师需要自己手工评估。*DOPCSM* 模型定义了评估报告的生成方法，即根据分析后的数据以及预先定义的评价标准自动生成评估报告。该过程将分析结果以一种直观的形式呈现给学生和教师，并且能够达到及时和持续生成的要求。

4 评估指标设计

为了能够通过一种定量的方法来评估学生的实践质量，本文根据现有的项目过程数据定义了一系列评估指标。这些评估指标不仅可以帮助教师评估项目的质量和学生的表现，同时还能够让学生认识到项目的问题，从而修复缺陷和提升质量，使项目开发过程形成一个良性循环。根据指标的性质，本文将这些指标分成两类：过程指标和代码指标。以下将详细介绍这两类指标。

4.1 过程指标

过程指标反映出项目生命周期的演进情况，能够帮助教师对项目形成一个宏观的认识，对学生的开发过程表现做出评估。另外，学生也可以通过对过程指标的持续监控和调整来提高项目的整体质量。

以下是具体的过程指标：

- **构建频率**是指一段时间内平均构建的时间间隔。本指标代表着项目团队在特定时间段内的开发表现，比如代码的更新速率。根据评估的情况差异，时间段可以相应的变化，例如最近一周的构建频率。本指标可以从持续集成服务器中直接获得或由计算得到。
- **构建状态**记录了每一次构建的基础信息，例如构建时间、构建结果、持续时间等。通过对构建状态的图表化，项目的构建细节可以被直观的展现。本指标描述了整个项目的构建概况。这些构建状态可以直接从持续集成服务器中获得。
- **构建成功率**是指一段时间内成功构建的占比。本指标直接体现了特定时间段内提交代码的整体质量。较低的构建成功率意味着代码中存在着大量的逻辑错误或者复杂的模块耦合等。本指标可以从持续集成服务器中直接获得或由计算得到。
- **部署状态**类似构建状态，反映出项目产品的部署表现。部署状态会受到构建状态的影响。本指标可以从持续部署服务器中直接获取。
- **部署成功率**类似构建成功率，直接体现了特定时间段内的部署质量。部署成功率会收到构建成功率的影响。本指标可以从持续部署服务器中直接获得或由计算得到。

4.2 代码指标

代码指标反映出项目代码的质量，能够帮助学生从不同方面了解项目代码的问题，进而提升项目整体质量。教师也可以借此评价学生的编码水平。

以下是具体的代码指标，这些指标数据由测试服务器计算得到。

- **质量阀**是项目测试的最终结果，例如通过、警告、不通过。本指标直接反映了项目代码的质量水平。具体来说，质量阀是由若干条预先定义好的条件组成的，且质量阀的条件可以分等级定义，例如新增缺陷数超过 50 个为不通过，超过 30 个但不超过 50 个为警告，小于等于 30 个为通过。质量阀的最终结果由所有条件中最差的级别决定的，比如有 3 个条件是通过的，2 个条件为警告级别，还有 1 个条件是不通过，则质量阀由不通过的条件决定，即最终不通过。
- **技术负债**代表开发人员需要修复所有问题需要花费的时间。本指标反映出代码的可维护性，若解决技术负债需要的时间越长，则代码中隐藏的问题就越严重。技术负债由不同等级问题数量和对应的单位修复时间决定的。
- **问题严重性**包含 5 个级别，分别是阻断(Blocker)、严重(Critical)、主要(Major)、次要(Minor)、信息(Info)。本指标计算各个级别包含的问题数量，问题的级别判定有测试服务器完成。通过问题级别的图表化，评估者可以清晰的认识到项目的问题级别分布情况。
- **问题类型**包含 3 个级别，分别是缺陷(Bug)、易损点(Vulnerability)、代码异味(Code smell)。缺陷是指任何与设计不符的系统行为问题；易损点是指非法使用系统导致的问题，大多与系统安全性有联系，一般是由于设计失误或实现错误引起的；代码异味是指隐藏在代码深层次的问题，可能在今后会演化成缺陷。通过问题类型的图表化，评估者可以清晰的认识到项目的问题类型分布情况。

- **复杂度**是由代码分支路径的数量计算得到。每当一个控制流函数产生一个分支，代码的总体复杂的就会增加 1，例如“if”分支会使复杂度增加 1。本指标反映出代码的可维护性和可读性。复杂度越高，项目代码质量可能会越糟糕。
- **文档百分比**代表注释行数占总代码行的比例。具体计算方法如下：

$$文档(\%)=\frac{注释行数}{(注释行数+代码行数)}*100$$

其中代码行数指物理代码数，不包含注释行。本指标反映出源码的可读性和可维护性。

- **重复百分比**代表重复代码行所占的比例，即重复代码行与总代码行的比值。如果重复百分比过高，则意味着源码中有可能存在代码异味。

所有的过程指标和代码指标并不是都适用于每个评估情形，因此教师需要根据实际情况挑选出合适的评估指标。举例来说，在学生项目进行的中期，教师可以仅仅只关注和监控项目的构建频率，以此来对学生团队的开发进度和过程质量有个初步的了解；而到了项目进度末，教师可以选择部署成功率来评估项目的可达性，也就是项目制品是否持续可用。另外，教师还可以挑选部分适合的代码指标来评价项目的源码质量。DevOps 教学的重点之一是通过学生对 DevOps 过程的体验，使他们感受 DevOps 对交付速度的提升以及对产品质量的保障。因此在 DevOps 教学实践中，过程指标比代码指标更为重要，实践过程的好坏会影响代码质量。一般来说，过程质量越高代码质量也不会很差。

5 模型原型设计

为了展示 *DOPCSM* 模型的实用性和可行性，本文实现了一个过程能力支持系统原型。该原型使用 *GitHub* 作为代码仓库，*Jenkins* 作为持续集成服务器，*SonarQube* 作为测试平台，*Maven* 作为构建工具以及 *Docker* 作为容器工具。接下来几章节从实际使用的流程角度出发阐述了模型的作用。

5.1 准备工作和项目创建

在项目开始前，每个学生需要在原型系统中注册一个账号。原型系统的管理员（通常是教师或助教）也可以在 DevOps 实践课程开始前统一配置学生账号。创建账号的过程会同步到 *Jenkins* 和 *SonarQube* 中，也就是一次注册会在各个系统中同步创建账号。原型系统的用户界面如图 3 所示。

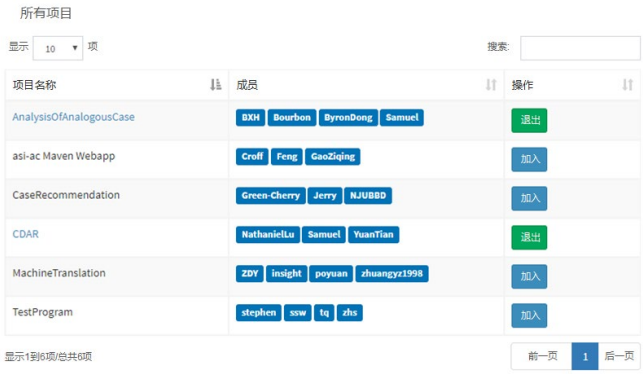

Fig. 3 Project List

图 3 项目列表

如果学生项目团队想要在实践过程中使用 *GitHub* 来管理源代码，他们则需要在 *GitHub* 中先创建一个代码仓库。创建成功之后，学生需要自己编写 *Jenkinsfile* 来定制项目的工作流，同时也需要编写 *SonarQube* 代码测试的属性文件 *sonar-project.properties*。这些文件在每个项目中都是不同的，且它们需要被放置在项目的根目录下。最后，学生需要在已经登录的情况下创建和初始化项目。

项目仓库需要填写学生项目团队的 *GitHub* 仓库 URL。如果(Web)项目在之前就已经构建过并且部署到服务器中了，项目制品一栏就可以填写制品的 URL。在项目创建的过程中，原型系统会通过 API 向 *Jenkins* 和 *SonarQube* 同步创建项目，同时会通过模板自动完成 *Jenkins* 项目的配置。最终，项目创建的信息会被存储到原型系统中。

创建好的项目会在项目列表中列出（图 3）。此列表展示了项目的基本信息，比如项目名和项目成员。详细信息可以通过点击项目名查看。

5.2 团队管理

通过“操作”列表中的按钮，学生可以更方便的加

入和退出项目团队。当学生没有加入项目时，按钮为“加入”；反之则是“退出”。另外，一个学生可以加入多个不同的项目团队。

5.3 项目测试

到这一步为止，项目的创建和初始化都已经完成了，之后学生应该关注项目的开发，以满足快速变更的需求。项目的源码应该被快速、频繁的提交，一天一次提交，甚至一天三次提交。每当 *GitHub* 中的代码发生了变更，*Jenkins* 会通过钩子(Hook)自动触发流水线，下载源码到本地工作区，然后交给 *SonarQube* 执行测试。原型系统会抽取测试结果数据，并以指标的形式展现出来 (图 4)。

界面上展示的指标已经在第 4 节中详细说明了。图中带符号的数字代表与之前一次分析的差异，正号表示增加，负号表示减少。除此以外，数字会用两种不同的颜色标注，绿色代表提升，红色代表恶化。环形图展示了各个级别的问题。为了更清晰的区分级别，原型系统使用不同的颜色表示程度，越偏向红色的代表越严重，越偏向绿色的表示程度越轻。界面中的柱状图展示了不同级别问题数量的变化情况以及同总问题数对比的情况。

Fig. 4 Test results

图 4 测试结果

5.4 项目构建

测试阶段之后是项目构建阶段。*Jenkins* 会调用项目的使用的构建工具（如 *Maven*）执行制品打包任务，最后生成可执行的制品（如 WAR 文件）。整个构建过程信息会展示在原型系统中，如图 5 所示。最近一次构建结果会在界面中展示。除此之外，原型系统还展示了多次构建的统计后结果，比如构建频率、特定时间段内的构建成功率和构建状态。

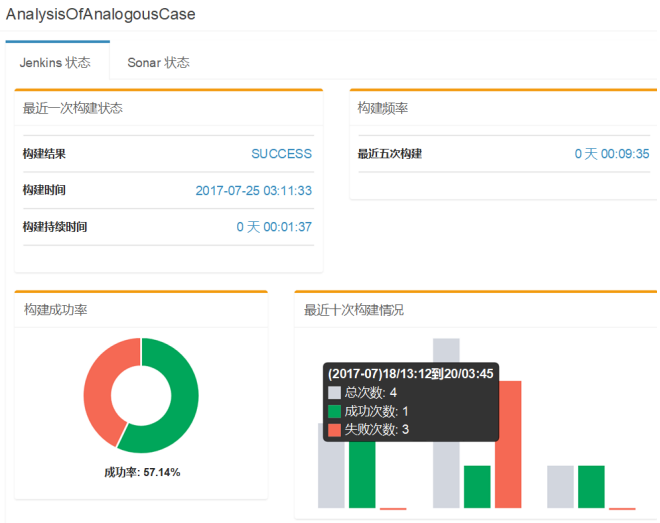

Fig. 5 Build result

图 5 构建结果

除了以上所述的项目数据，原型系统还提供了个人过程数据。如图 6 所展示的，用户引入的问题会根据严重性分组，而且也会根据此用户参加的项目分组。同样的，图表也使用不同颜色来区分程度。

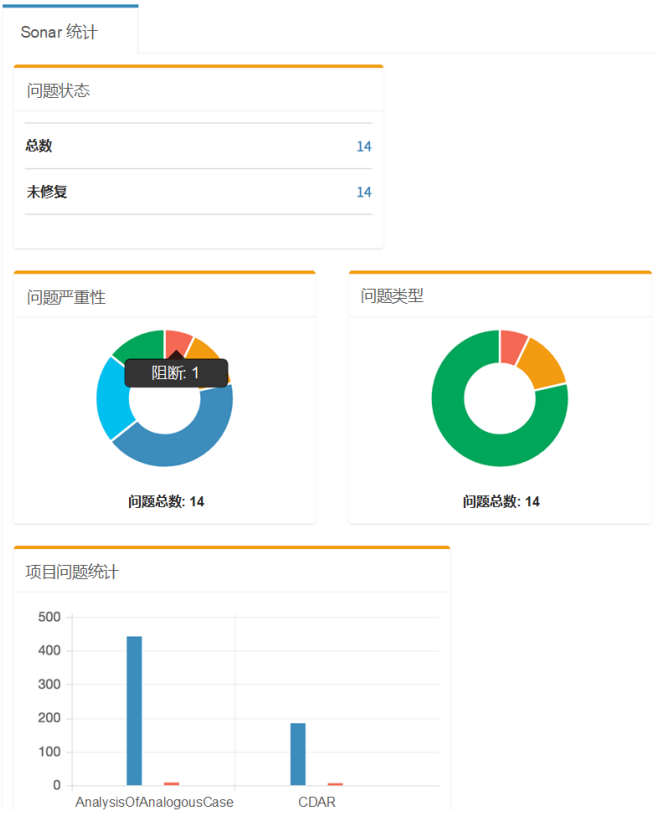

Fig. 6 User status

图 6 用户数据

以上阐述的项目工作流是 DevOps 实践的第一次循环。之后学生只需要专注于代码的编写，并根据系统页面中持续反馈的分析结果改进源码。在实践的末期，教师可以根据分析的结果数据轻松的评估项目团队和项目成员的过程管理及开发表现。

5.5 实际应用案例

我们已经将本原型系统应用在南京大学的教学中。图 3 展示了学生在 DevOps 课程实践中创建的真实项目。通过本原型系统，学生就不需要在自己的设备上进行繁琐的开发环境配置。另外，本原型系统也减少了学生在使用开发环境中遇到的问题，同时也节省了教师和助教帮助学生解决这类问题的时间。

在 DevOps 课程实践中，本原型系统被证实有助于收集过程数据。图 4 和图 5 分别展示了其中一组学生团队开发项目的测试结果和构建结果。当学生进行开发活动时，其产生的过程数据是可以被收集的。在课程结束时，这些数据会被用于学生团队的表现评估。总体来说，本原型系统满足了课程实践的初衷——简化开发环境的配置和使用，以及更方便的收集过程数据。*DOPCSM* 原型系统构建了一个易用的开发环境来帮助学生实践 DevOps，在保障代码质量的同时提高代码构建频率。

6 讨论

本文提出了一个通用的过程能力支持模型，使用者可以根据实际应用场景实现自己的教学系统。*DOPCSM* 模型旨在帮助学生管理 DevOps 实践，同时辅助教师监控和评估学生的实践过程。学生可以通过本模型的支持系统与其他开发人员协作，可以根据持续反馈的结果不断提升项目质量。教师能够监控学生项目进展并且评估项目团队的表现。在当前阶段，*DOPCSM* 模型满足我们开展 DevOps 实践课程的需求和预期。然而，在将本模型应用到其他教学场景时，我们也发现了一些问题，在这里有必要说明。

6.1 工作流

当前情况下 *DOPCSM* 模型的工作流是固定的，这个工作流是基于软件开发生命周期简化而成的。但是，当教学环境发生改变时，工作流也许就不能适用于所有情况了。

除此之外，模型中每个阶段的工具都是需要用户自

行定义和配置的，这就对教师提出了更高的要求。学生实践的代价变小了，但是教师初次搭建和配置环境的工作量就增加了。

6.2 指标定义

目前，本文提供了若干指标来帮助教师和学生评估工作质量。可是现有的评估指标集合是基于已经存在的工具的，并没有做长远的考虑。随着 DevOps 教学经验的增长、理解的深入，我们或许需要从不同角度获得更多的评估指标。从这个方面来说，*DOPCSM* 模型中的自动评估模块需要加强，以便支持更多的指标。同时还需要集成不同种类的算法来从不同角度对学生进行计分，比如生产率、贡献率等。

7 结束语

由于缺少工具的支持，DevOps 教学的实施会变得尤其艰难。为了帮助学生和教师更好的开展 DevOps 实践，本文提出了一种过程能力支持模型。此模型定义了实践必要的环境，过程管理的方法以及项目评估的手段。教师可以根据现实的教学需求自行实现 *DOPCSM* 模型的支持系统。在南京大学开展的试验中，该模型取得了积极正面的结果。*DOPCSM* 模型不仅能够让学生关注 DevOps 过程方法而不仅仅是工具和技术，而且还可以减轻教师评估项目的负担。

在未来的工作中，我们将改进 *DOPCSM* 模型，使之满足更多的教学需求。首先，评估的指标需要改进和完善，并且经过理论的验证；其次，工作流将提供定制化的解决方案，且模型需要有更丰富的功能来应对更复杂的教学环境。

References

[1] C. Ebert, G. Gallardo, J. Hernantes, et al. DevOps[J]. IEEE Software, 2016, 33(3): 94–100.

[2] M. Huttermann. DevOps for Developers[M]. Expert's Voice in Web Development. New York: Apress, 2012.

[3] L. E. Lwakatare, P. Kuvaja, M. Oivo. Dimensions of DevOps[C]. Proceedings of the 16th International Conference on Agile Software Development (XP '15), Helsinki, Finland, 2015. Cham: Springer, pp. 212–217.

[4] F. Erich, C. Amrit, M. Daneva. A Mapping Study on Cooperation between Information System Development and Operations[C]. Proceedings of the 15th International Conference on Product-Focused Software Process Improvement (PROFES '14), Helsinki, Finland, 2014. Cham: Springer International Publishing, pp. 277–280.

[5] S. K. Bang, S. Chung, Y. Choh, et al. A Grounded Theory Analysis of Modern Web Applications: Knowledge, Skills, and Abilities for DevOps[C]. Proceedings of the 2nd Annual Conference on Research in Information Technology (RIIT '13), Orlando, Florida, USA, 2013. New York, NY, USA: ACM Press, pp. 61–62.

[6] D. G. Feitelson, E. Frachtenberg, K. L. Beck. Development and Deployment at Facebook[J]. IEEE Internet Computing, 2013, 17(4): 8–17.

[7] IBM. Delivery pipeline[OL]. [2017-07-20]. https://console.ng.bluemix.net/catalog/services/delivery-pipeline.

[8] AWS. AWS codePipeline[OL]. [2017-07-20]. https://aws.amazon.com/codepipeline/.

[9] Microsoft Azure. Visual Studio Team Services[OL]. [2017-07-20]. https://azure.microsoft.com/en-us/services/visual-studio-team-services/.

[10] DevOpsGuys. Educate - UK DevOps training courses and programmes[OL]. [2017-07-20]. https://www.devopsguys.com/devops-educate/.

[11] DevOps Institute. Setting the standard in DevOps training[OL]. [2017-07-20]. http://devopsinstitute.com/.

[12] M. Airaj. Enable cloud DevOps approach for industry and higher education[J]. Concurrency and Computation: Practice and Experience, 2017, 29(5).

[13] S. Krusche, L. Alperowitz. Introduction of continuous delivery in multi-customer project courses[C]. Companion Proceedings of the 36th International Conference on Software Engineering (ICSE Companion '14), Hyderabad, India, 2014. New York, NY, USA: ACM, pp. 335–343.

[14] H. B. Christensen. Teaching DevOps and cloud computing using a cognitive apprenticeship and story-telling approach[C]. Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16), Arequipa, Peru, 2016. New York, NY, USA: ACM, pp. 174–179.

[15] J. C. Adams. Chance-It: An object-oriented capstone project for CS-1[J]. SIGCSE Bull., 1998, 30(1): 10–14.

[16] J. O. Hamblen, H. L. Owen, S. Yalamanchili, et al. An undergraduate computer engineering rapid systems prototyping design laboratory[J]. IEEE Transactions on Education, 1999, 42(1): 8–14.

[17] D. A. Umphress, T. D. Hendrix, J. H. Cross. Software process in the classroom: The capstone project experience[J]. IEEE Software, 2002, 19(5): 78–81.

[18] D. P. Groth, E. L. Robertson. It's all about process: Project-oriented teaching of software engineering[C]. Proceedings of the 14th Conference on Software Engineering Education and Training (CSEET '01), Charlotte, NC, USA, USA, 2001. IEEE, pp. 7–17.

[19] B. R. von Konsky, M. Robey. A case study: GQM and TSP in a software engineering capstone project[C]. Proceedings of the 18th Conference on Software Engineering Education Training (CSEET '05), Ottawa, Ont., Canada, 2005. IEEE, pp. 215–222.

[20] S. Jarzabek, P.-K. Eng. Teaching an advanced design, team-oriented software project course[C]. Proceedings of the 18th Conference on Software Engineering Education Training (CSEET '05), Ottawa, Ont., Canada, 2005. IEEE, pp. 223–230.

[21] M. Raza, J. ao Pascoal Faria, R. Salazar. Helping software engineering students analyzing their performance data: Tool support in an educational environment[C]. Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C '17), Buenos Aires, Argentina, 2017.

Piscataway, NJ, USA: IEEE Press, pp. 241–243.
[22] W. S. Humphrey. PSP^{SM}: A Self-Improvement Process for Software Engineers. Addison-Wesley Professional, 2005.
[23] G. Rong, H. Zhang, S. Qi, et al. Can software engineering students program defect-free?: An educational approach[C]. Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16), Austin, Texas, 2016. New York, NY, USA: ACM, pp. 364–373.
[24] W. S. Humphrey. Introduction to the Team Software Process. AddisonWesley Professional, 2000.
[25] M. Ohtsuki, K. Ohta, T. Kakeshita. Software engineer education support system ALECSS utilizing DevOps tools[C]. Proceedings of the 18th International Conference on Information Integration and Webbased Applications and Services (iiWAS '16), Singapore, Singapore, 2016. New York, NY, USA: ACM, pp. 209–213.
[26] B. P. Eddy, N. Wilde, N. A. Cooper, et al. CDEP: Continuous delivery educational pipeline[C]. Proceedings of the SouthEast Conference (ACM SE '17), Kennesaw, GA, USA, 2017. New York, NY, USA: ACM, pp. 55–62.
[27] V. Armenise. Continuous delivery with Jenkins: Jenkins solutions to implement continuous delivery[C]. 2015 IEEE/ACM 3rd International Workshop on Release Engineering (RELENG '15), Florence, Italy, 2015. IEEE, pp. 24–27.
[28] M. de Bayser, L. G. Azevedo, R. Cerqueira. ResearchOps: The case for DevOps in scientific applications[C]. 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM '15), Ottawa, ON, Canada, 2015. IEEE, pp. 1398–1404.
[29] A. A. Mohallel, J. M. Bass, A. Dehghantaha. Experimenting with Docker: Linux container and BaseOS attack surfaces[C]. 2016 International Conference on Information Society (i-Society '16), Dublin, Ireland, 2016. IEEE, pp. 17–21.
[30] M. A. R, J. K. Patel, S. Akhtar, et al. Docker container security via heuristics-based multilateral security-conceptual and pragmatic study[C]. 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT '16), Nagercoil, India, 2016. IEEE, pp. 1–14.

GU Shenghui was born in 1995. He is a M.S. student at Nanjing University. His research interests include DevOps, software architecture, etc.

顾胜晖(1995-)，男，江苏苏州人，南京大学硕士研究生，主要研究领域为 DevOps 和软件架构。

ZHANG He received the Ph.D. degree from The University of New South Wales. He is a Full Professor of Software Engineering at the Nanjing University and a Senior Member of China Computer Federation (CCF). He has published over 100 peer-reviewed papers in high quality international conferences and journals, and won 9 Best Paper Awards from several prestigious international conferences (e.g., ICSE, ESEM, ICSP/ICSSP, APSEC, and EASE). His research interests include Software & Systems Engineering Process, Software Architecture, DevOps and Container, Empirical Software Engineering, etc.

张贺，男，于新南威尔士大学获得博士学位，现为南京大学软件工程教授，CCF 高级会员。近年来，著有英文专著两部，并在包括国际软件工程大会（ICSE）在内的国际重要软件工程期刊和会议上发表论文 100 余篇，其中 9 篇会议长文获最佳论文奖。主要研究领域为软件及系统工程过程，软件架构，开发运维与容器技术，经验软件工程等。

SHAO Dong was born in 1976. He received the M.S. degree in Department of Computer Science from University of Toronto, Canada in 2003. He is an associate professor at Nanjing University. His research interests include software engineering, software process, agile software development, etc.

邵栋(1976-)，男，安徽省淮南市人, 2003 年加拿大多伦多大学计算机系科学硕士，南京大学副教授，主要研究领域为软件工程，软件过程，敏捷软件开发。