RICK：从生产环境数据生成模拟对象

Deepika Tiwari、Martin Monperrus、Benoit Baudry 瑞典皇家理工学院，瑞典{deepikat、monperrus、baudry}@kth.se

arXiv:2302.04547v1 [cs.SE] 9 Feb 2023

摘要—测试替身（如模拟对象和桩）是单元测试中非常实用的固定装置。它们使开发人员能够将各个组件与系统内部或外部的其他组件隔离开来进行测试。然而，在测试中实现测试替身并不简单。在本次演示中，我们介绍 RICK，这是一种能够观测运行中的应用程序，从而自动生成包含逼真模拟对象和桩的测试的工具。RICK 监控目标方法的调用及其与外部组件的交互。基于从这些观测结果中收集的数据，RICK 生成包含模拟对象、桩和基于模拟对象的预言机的单元测试。我们将展示 RICK 的功能，以及如何将其用于真实世界的 Java 应用程序，以生成包含模拟对象的测试。

索引词—**模拟对象、桩、生产环境、预言机、测试工具**

I. 引言

软件应用程序由多个相互耦合、共同实现整体目标的系统组成。这有助于渐进式和模块化开发，也便于复用外部库。然而，为了单独测试每个组件，必须将其与其他组件解耦。解决这一问题的一种方法是在单元测试中使用测试替身。外部组件会被替换为称为 模拟对象 [1]的骨架实现。由于模拟对象替代了实际组件，与该组件的所有交互都会被绕过。相反，模拟对象的行为通过 桩来声明，因此，当使用特定输入调用模拟对象的方法时，它会返回预设响应。模拟对象和桩有助于更专注地验证被测组件的正确性、减少副作用、加快测试执行，并更快地进行故障定位 [2]。然而，使用它们并非易事。编写单元测试时，开发人员必须手动决定哪些组件可以被模拟，以及如何为它们设置桩 [3]。

通过基于搜索的算法 [4] 或符号执行 [5] 自动生成的模拟对象可能无法反映真实的生产环境行为。我们的核心见解是，利用生产环境中收集的数据生成带有模拟对象的测试，可以确保对实际用户触发的行为进行测试，甚至还能补充开发人员编写的测试套件 [6]。我们提出了 RICK，这是一种用于观察运行中的应用程序并收集数据的工具，旨在自动生成使用模拟对象的测试，并根据生产环境调用对其行为进行桩替代。此外，RICK 生成的测试包含基于模拟对象的预言机，这些预言机源自生产环境中收集的数据，用于验证目标方法调用及其与外部组件交互的不同切面。

本演示将概述 RICK。我们的配套视频 [7] 演示了 RICK 与名为 GraphHopper 的开源路由应用程序 [8]协同运行的效果。

```
1类 ClassUnderTest {2    ExtTypeOne extField;3    ...4 整数
```

清单 1：一个被测方法，其中包含两个可模拟方法调用，RICK 将其确定为测试生成目标

II. RICK

本节重点介绍 RICK 的关键阶段，并展示生成的测试示例。我们还概述了如何将 RICK 用于实际的 Java 应用程序。

A. 模拟对象生成的关键阶段

RICK 分三个阶段生成测试和模拟对象。首先，RICK 确定一组可配置的目标方法，用于测试生成。我们将每个目标方法称为被测方法（*MUT*）。MUT 包含对外部类型对象的可模拟方法调用；这些外部类型可以定义为 MUT 声明类型中的字段，也可以是 MUT 的参数。例如，清单 1 展示了在 ClassUnderTest 类中定义的一个名为 methodUnderTest 的 MUT。该 MUT 中有两个可模拟方法调用：一个针对字段 extField（第 6 行），另一个针对参数 extParam（第 7 行）。RICK 的第二个阶段发生在应用程序处于生产环境时。RICK 会收集与每次 MUT 调用对应的数据，以及 MUT 内发生的可模拟方法调用的数据。对于 MUT，收集的数据包括接收对象（即调用 MUT 的对象）、传入调用的参数，以及调用返回的对象。对于可模拟方法的调用，RICK 会捕获参数和返回对象。最后，在第三个阶段，RICK 离线使用收集到的数据为 MUT 生成测试。在测试中，外部对象会被模拟对象替代。可模拟方法调用会变成模拟方法调用，即在模拟对象上调用，而非实际对象。它们的行为会根据生产环境中获得的观测结果进行桩设。RICK 会生成三种不同类型的基于模拟对象的预言机，以验证生产环境观测结果。

B. 生成的测试

RICK 生成的测试通过模拟对象、桩和基于模拟对象的预言机，模拟在生产环境中进行的观测结果。

```
@Test
public void 被测方法测试() {//
    准备
// 从序列化的生产数据重建对象
ClassUnderTest生产环境对象 =反序列化(new File(
        "receiving.xml"));
// 注入模拟字段
ExtTypeOne 模拟外部字段 =
        在被测类中注入模拟字段 外部字段 ();
// 模拟参数
ExtTypeTwo 模拟外部参数 = 模拟对象(ExtTypeTwo.类);
// 桩行为
当(模拟外部字段.可模拟方法一(42)).返回(99);
当(mockExtParam.mockableMethodTwo(27)).返回(17);

// 执行阶段
int实际值 = 生产环境对象.methodUnderTest(64,
        mockExtParam);

// 断言
...
```

图 2：RICK 测试的准备和执行阶段

```
assertEquals(42, actual);}
```

清单 3：带有输出预言机的 RICK 测试的断言阶段

对于被测方法的每次调用，RICK 都会生成三个测试，每个测试都包含一个预言机，专门关注该次调用的不同方面。三个生成的测试具有相同的准备和执行阶段。对于清单 1中的被测方法，这些阶段如清单 2所示。接收对象会根据捕获的生产快照进行反序列化并重新创建（第 5 行），外部对象则会被替换为模拟对象（第 6 至 9 行）。随后，根据生产环境中的观测结果为可模拟方法设置桩（第 10 至 12 行）。接着调用被测方法（第 14 行）。带有输出预言机的测试中的断言阶段，会验证被测方法的输出是否与生产环境中观测到的结果一致。该阶段见清单 3第 16 行。清单 4所示的、带有参数预言机的测试，其断言阶段会验证测试中可模拟方法调用所使用的参数是否与生产环境中的一致。对于带有调用预言机的测试，断言阶段（清单 5）会验证模拟方法调用的顺序和次数是否与 RICK 在生产环境中观测到的一致。这三类基于模拟对象的预言机确保测试能够重现生产环境条件，包括生产环境中的参数和响应，以及与外部组件的交互。

捕获方法调用、持久化相关对象快照，并通过预言机表达这些信息，并非易事。我们也意识到，正确生成并保存到磁盘的测试，并不能保证忠实地复现生产环境上下文。反序列化后，对象可能无法恢复到生产环境中的状态；引入模拟对象也可能改变被测方法的行为。开发 RICK 并将其用于复杂的真实项目，有助于我们深入理解软件的动态行为。

C. 用法

RICK 是开源的，可以从 GitHub 上的代码库克隆 [9]。README 和代码库维基提供了更多信-

```
验证(mockExtField,至少一次()).可模拟方法一(42); 验证(mockExtParam,至
```

清单 4：使用参数预言机的 RICK 测试的断言阶段

```
InOrder orderVerifier =inOrder(mockExtField,
```

清单 5：使用调用预言机的 RICK 测试的断言阶段

有关如何在 Java 应用程序中使用 RICK 的详尽说明及示例。我们附带的视频 [7] 展示了 RICK 在开源路由应用程序 GraphHopper [8]中的实际运行。RICK 由三个核心模块组成，每个模块都通过简单的命令调用。它们的目标可概括如下：首先，给定 Java 项目的路径，选择模块会识别其中的被测方法及相应的可模拟方法。开发人员可以配置生成的列表。接下来，对于每个被测方法及其可模拟方法，插桩模块会生成切面，用于定义在生产环境中遇到相应调用时应执行的指令。这些切面会打包到一个 JAR 包中。只需将此 JAR 包作为 Java 代理附加，并像往常一样调用应用程序即可。捕获到的被测方法及可模拟方法的调用数据会被序列化并保存到磁盘。最后，开发人员触发生成模块后，该模块会处理生产环境中每个被测方法所收集的数据，并生成 JUnit 测试 [10], , 其中包含通过 Mockito [11], [12]定义的模拟对象、桩和基于模拟对象的预言机。

D. 预期使用场景

RICK 基于生产环境观测结果生成测试，因此可以独立于应用当前的测试状态使用。事实上，如果应用尚无测试套件，RICK 生成的测试可以用来搭建一个测试套件。另一方面，如果应用已有开发人员编写的测试套件，RICK 生成的测试可以补充反映生产环境中实际使用情况的行为 [13]。此外，开发人员可以根据自定义标准（例如覆盖率或变异测试）选择被测方法 [6], , 也可以将目标设为近期修改过的方法。RICK 生成的测试通过捕获的对象和交互来模拟生产环境行为，并可用作回归测试。

III. 结论

RICK 是一种自动化测试生成工具。它的创新之处在于利用从运行中的应用程序中收集的数据，通过模拟对象、桩和基于模拟对象的预言机，生成能够模仿生产环境行为的测试。本演示将介绍 RICK 的关键阶段、使用方法和应用案例。未来的一个重要方向是扩展 RICK，使其采用更细粒度的监控和快照技术，用于生成测试和模拟对象。有关 RICK 的更多详情，包括在真实项目中的评估结果，请参见 [14]。

参考文献

[1] T. Mackinnon、S. Freeman 和 P. Craig，《Endo-testing：使用模拟对象进行单元测试》，*Extreme programming examined*，第 287–301 页，2000 年。[2] D. Thomas 和 A. Hunt，《模拟对象》，*IEEE Software*，第 19 卷，第 3 期，第 22–24 页，2002 年。[3] E.T.Barr、M.Harman、P.McMinn、M.Shahbaz 和 S. Yoo，《软件测试中的预言机问题：综述》，*IEEE transactions on software engineering*，第 41 卷，第 5 期，第 507–525 页，2014 年。[4] A. Arcuri、G. Fraser 和 J. P. Galeotti，《为自动化单元测试生成 TCP/UDP 网络数据》，载于 *Proceedings of the 2015 10*2015 年，*th Joint Meeting on Foundations of Software Engineering*，第 155–165 页。[5] M. Islam 和 C. Csallner，《Dsc+ mock：支持面向接口编程的测试用例+ 模拟类生成器》，载于*Proceedings of the Eighth International Workshop on Dynamic Analysis*，2010 年，第 26–31 页。[6] D. Tiwari、L. Zhang、M. Monperrus 和 B. Baudry，《生产环境监控以改进测试套件》，*IEEE Transactions on Reliability*，第 1–17 页，2021 年。

[7] “Rick：从生产环境数据生成模拟对象，” https://youtu.be/ljuUfbC-IZw/，访问日期：2023-01-06。 [8] “Graphhopper，” https://www.graphhopper.com/，访问日期：2022-12- 28。[9]“Rick，” https://github.com/castor-software/pankti#rick，访问日期：2022-12-28。[10] “Junit 5，” https://junit.org/junit5/，访问日期：2022-12-28。 [11] “Mockito，” https://site.mockito.org/，访问日期：2022-12-28。 [12] S. Mostafa 和 X. Wang，《软件测试中模拟框架使用情况的实证研究》，载于*2014*年第*14*届国际软件质量会议，2014年，第127–132页。 [13] Q. Wang、Y. Brun 和 A. Orso，《行为执行比较：测试是否能代表实际环境行为? 》，载于*2017*年 *IEEE* 国际软件测试、验证与确认会议（*ICST*）。IEEE，2017年，第321–332页。 [14] D. Tiwari、M. Monperrus 和 B. Baudry，《使用生成的模拟对象模仿生产环境行为》，*arXiv* 预印本 *arXiv:2208.01321*，2022年。