Code Critters：基于代码块的测试游戏

Philipp Straubinger帕绍大学帕绍，德国

Laura Caspari帕绍大学帕绍，德国

Gordon Fraser帕绍大学帕绍，德国

arXiv:2304.02246v1 [cs.SE] 5 Apr 2023

摘要——**在学校、高等教育和个人学习中，学习编程已十分普遍。尽管测试是编程的重要组成部分，但由于人们认为时间和知识不足，或仅仅因为觉得测试不那么重要或有趣，它在教育中常常被忽视。为了让测试更具吸引力，我们推出了*Code Critters*，这是一款基于测试概念的塔防游戏：游戏的目标是沿着小动物从家园前往塔的路径放置魔法地雷，使地雷能够区分执行正确代码的小动物和执行含缺陷代码的小动物。游戏中的代码使用基于代码块的语言显示和编辑，让年纪较小的学习者也能轻松上手。地雷编码了测试输入和测试预言机，从而使测试成为游戏中不可或缺且有趣的一部分。**

索引词—**游戏化、变异、基于代码块、软件测试、教育、严肃游戏**

I. 引言

编程已成为学校教育 [1]以及高等教育中的常见内容，同时也是业界备受追捧的一项技能 [2]。尽管软件测试在实践中是软件开发不可或缺的一部分，但它在编程教育中却经常被忽视 [3],，尽管高等教育领域对此的认识日益提高，甚至有人提议将测试纳入学校课程 [4]–[6]。这通常是由于人们缺乏测试教学方面的意识和技能，同时学习者和程序员普遍认为测试是一项枯燥乏味的任务。只要软件测试尚未以更具吸引力且更易于理解的方式进行教学，这种情况就不太可能改变。

为了以更具吸引力的方式向学习者传授测试知识，我们介绍了*CodeCritters*这款严肃游戏。它以塔防为基础，主要目的并非娱乐，而是教育。图 1 展示了游戏棋盘的运行状态：玩家必须从变异体手中拯救类人小动物（例如图 1 右侧所示的小动物）。变异体是带有类似僵尸的绿色头部、位于更左侧的小动物，玩家可以根据其行为来区分它们，而这些行为被编码在易于理解的短小基于代码块的代码片段中。游戏通过放置魔法地雷来进行（例如图 1 中泥地路径的起点处），这些地雷本质上就是测试用例，放置在小动物从左侧的家前往图 1 右侧目标塔的路径上。地雷既表示测试输入（例如坐标、地形类型），也表示以基于代码块的代码片段呈现的测试预言机。通过构建基于不同软件概念的适当关卡和变异体，可以将各种软件概念融入游戏玩法中。



图 1：游戏进行期间的游戏棋盘

II. 背景

尽管软件测试十分重要，且高等教育领域对其关注度不断提高，但软件测试在编程教育中仍未得到充分体现 [7]。激励开发者编写测试的一种很有前景的解决方案是游戏化（例如， [8]），即将排行榜、积分或挑战等游戏元素应用于非游戏情境 [9]。严肃游戏将这一方法进一步推进，因为这类游戏是专门为训练、教育或模拟而制作的，玩家可以从嵌入其中的主题信息中学习，同时不会产生正在学习或工作的感觉 [10]。严肃游戏通常会改编广为人知的游戏类型，例如塔防，同时改变其中的某些方面以满足既定目标。令人意外的是，目前针对软件测试提出的严肃游戏仍然寥寥无几 [11]–[13]。

一种特别适合游戏化和严肃游戏的测试概念是变异测试，例如，它已成功融入 Code Defenders 游戏 [14]。变异测试包括在被测试的代码中插入人工缺陷，以识别并修复现有测试中的薄弱环节 [15]。在 Code Defenders 中，这一过程被游戏化为攻击者创建人工缺陷（变异体），而防御者则通过编写测试来检测这些缺陷。然而，Code Defenders 及其他将测试游戏化的尝试存在一个主要缺点：它们要求学习者具备相当高级的编程技能，因此更适合高等教育。为了让编程经验较少的学习者也能参与其中，必须将测试更早地融入编程教育中。

让年幼的学习者更容易接触编程的一种常见方法，是使用 Scratch 等基于积木的语言来呈现代码 [16]。学习者无需将代码作为文本输入，而是通过拖放预先定义的代码块，将它们排列到合适的位置，从而快速创建有趣的游戏和程序。鉴于基于积木的编程 [17],取得了成功，本文将探讨以类似方式降低软件测试的入门门槛，即采用基于积木的编程方法。

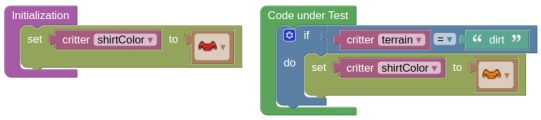


图 2：受测小生物

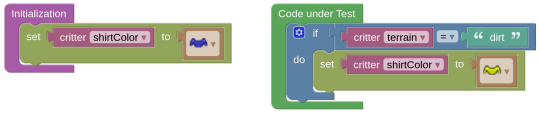


图 3：受测小生物的一个变异体

III. CODE CRITTERS

Code Critters 讲述了小动物们的故事：他们生活在森林中一个未知的栖息地，原本和平安宁。不幸的是，一场疾病暴发导致许多小动物变成了变异体。这些变异体的行为与其他小动物不同，并且正在摧毁栖息地，迫使健康的小动物逃往森林另一端的一座塔。在前往塔的途中，玩家必须放置魔法地雷，用来检查小动物的行为，并且只让健康的小动物通过。

A. 游戏概念

Code Critters 将变异测试和基于积木的编程融入广为人知的塔防游戏中。在经典的塔防游戏中，玩家必须沿着敌人前往游戏棋盘上某个特定位置的路径布置炮塔。敌人在抵达该位置的途中被消灭得越多，玩家最终获得的积分就越多。与传统塔防游戏不同，在*CodeCritters* 中，不仅有敌人，还有需要受到保护的平民。小动物的行为由受测小生物（CUT）表示，具体形式是一段简短的基于代码块的代码片段（图 2）。敌人（变异体）是通过对 CUT 进行变异而产生的（图 3）。为了避免使用击杀敌人这一较为暴力的概念，*Code Critters* 将炮塔替换为代表测试用例的地雷；这些地雷利用魔法困住变异体，而不是向它们开火。

B. 游戏机制

游戏棋盘（图 4）同时代表栖息地（生成点）和塔（目的地），其中包括连接两者的泥泞路径及其周围的森林。游戏棋盘由 256 个方块组成，使用从 1 到 16 的 x 和 y 坐标表示。每个方块只能具有一种特定的纹理，即草地、泥地、水、冰或木材，而小动物只能在草地、泥地和冰上行走。此外，地雷只能放置在小动物可以行走的方块上，这也是玩家在开始游戏之前必须进行的第一步操作。

小动物的行为由简短的代码片段描述；在小动物探索游戏棋盘时，这些代码片段会在循环中持续执行。被测代码在概念上类似于面向对象类，由两部分组成：初始化和受测代码。初始化类似于构造函数，用于定义小动物的初始属性值，例如

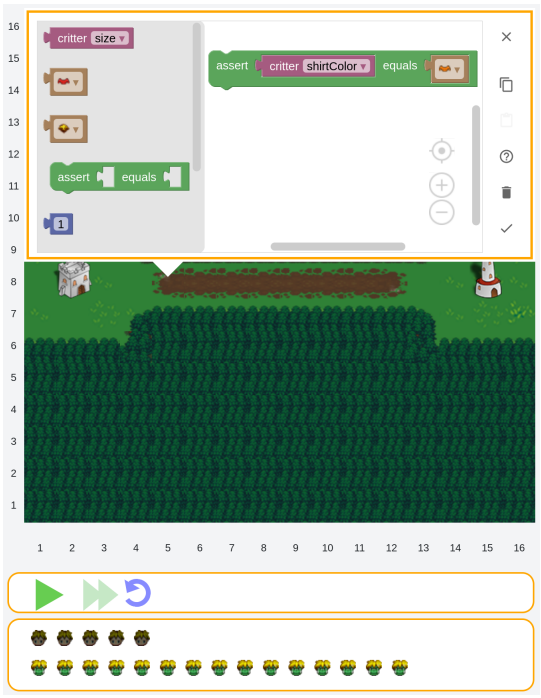


图 4：*CodeCritters* 的游戏棋盘

小动物的属性，例如它们的衬衫颜色或头发颜色。受测代码类似于一个方法，在小动物行走时于循环中持续调用，并接收纹理和坐标作为输入。图 2 所示的被测代码展示了一个健康小动物的简单示例：它初始化时穿着红色衬衫，走过泥土地格时颜色会发生变化。

变异生物包含一个或多个代码突变，例如图 3 中所示的不正确的衬衫颜色初始化和错误的衬衫颜色选择。游戏的目标是让健康的小动物到达并进入塔，同时利用地雷阻挡并困住变异体。

地雷代表测试用例：通过点击一个可通行的方块，可以在恰好一个方块上放置一枚地雷；这些可通行的方块通过其坐标和纹理表示测试输入。点击方块会打开一个对话框（图 4），玩家可以在其中针对给定的输入位置（方块）为被测代码创建新的测试。玩家可以使用与表示被测代码或变异体的相同基于代码块的语言，为地雷实现断言。图 4 对话框左侧的可用积木块可以通过拖放放置到右侧，并组合成一个测试用例。每个测试的主要积木块是断言块，它分为要检查的属性以及该属性在此时应具有的值。属性包括小动物的衬衫颜色、头发颜色和大小。衬衫颜色和头发颜色属于枚举类型，而大小属性是整数，可以检查其具体值，也可以检查它是否为偶数、奇数、负数、正数或素数值。还可以将基本信息存储到变量中并进行检查，以及对数值执行基本的数学运算。地雷所表示的测试会在游戏进行期间执行，针对任何踩到地雷的小动物进行检查。



图 5：完成当前关卡后的计分板

要捕获图 3 中的变异体，必须沿路径放置两枚不同的地雷来检测它：第一枚放置在某个草地格上，该处覆盖了初始化过程中发生的变异；第二枚放置在一个泥土地格上，以发现 if 条件中的变异（见图 1）。两枚地雷都必须断言衬衫颜色的预期值。这些地雷与变异体结合后，还可以作为抽象层来生成测试套件。游戏的次要目标是尽可能少地使用地雷来检测所有变异体，换句话说，就是尽量缩小测试套件。通常，地雷数量没有限制，但使用过多地雷会被扣除积分。

玩家在游戏棋盘上放置所有必要的地雷后，即可开始游戏（图 1）。此时，小动物会从左侧的栖息地出发，随机选择一条路径前往塔。游戏运行过程中，玩家可以随时暂停、重置或加速游戏。有效的地雷允许健康的小动物安全通过，而当测试失败时，则会困住变异体（参见第 III-C 节）。因此，无效的断言可能导致假阳性，困住健康的小动物而非变异体，从而使玩家损失积分。屏幕底部（图 4）会显示小动物和变异体，其中，到达塔的小动物将被标记为已获救，而被困住的则会显示为灰色。当最后一只未被困住的小动物或变异体到达塔时，游戏结束，并显示计分板（图 5）。计分板包含获救小动物和检测到的变异体的信息，以及使用的地雷数量和获得的时间奖励。游戏结束后，玩家可以查看变异体，以深入了解被测代码（CUT）是如何被变异的。玩家获得的分数会与之前游戏中获得的积分累加，并显示在*Code Critters*起始页面的公开排行榜上（图 6）。

C. 关卡

*Code Critters*提供了不同的关卡，并将其划分为教程、初级和高级类别（图 6）。我们当前的概念验证版本集成了十个关卡，但*CodeCritters* 还提供了一个关卡编辑器（图 7），用于创建新关卡。关卡编辑器允许用户创建包含五种不同纹理类型以及起点和终点的自定义地图。每个关卡都需要创建一个被测代码（CUT），并根据该被测代码创建一个或多个变异体。此外，小动物的数量、数量

[1]https://developers.google.com/blockly

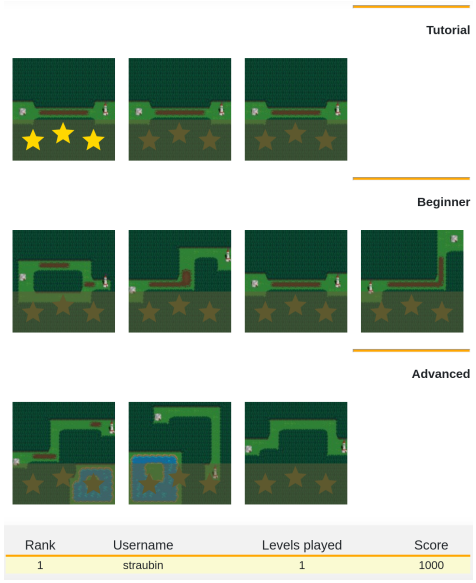


图 6：带有排行榜的 *Code Critters* 中的不同关卡



图 7：*Code Critters* 的关卡编辑器

玩家可以使用且不会扣分的地雷数量以及难度等级必须设置。

由于每个关卡都基于一个被测代码，因此可以通过选择合适的被测代码来调整难度和学习目标。玩家需要考虑被测代码的以下部分，并为它们编写测试，以捕获所有变异体：

- **初始化**：小动物初始配置中的变异
- **赋值**：所有属性或变量赋值的更改
- **分支**：if-else 子句的删除或更改
- **条件**：删除或更改 if-else- 子句中的条件

通过调整这些部分，关卡可以体现不同的测试概念。例如，使用一组变异体，其中每个变异体都包含一条或多条发生变化的赋值语句，可以教授完全语句覆盖；改变 if-else 子句的顺序或内容，则可以教授分支覆盖。改变甚至移除合取或析取中的条件，有助于理解条件覆盖。

不仅可以通过多种方式对被测代码进行变异以增加难度，还可以修改和设计游戏棋盘本身来进一步提高难度。例如，可以调整游戏棋盘，为小动物和变异体提供多条通往塔的路径。这就要求针对不同轨道设置不同的地雷，因为方块的纹理和坐标各不相同；某些地雷可能只在其中一条路径上独有，而其他地雷则必须添加到每条路径中，以确保捕获所有变异体，从而实质上帮助学习者理解输入划分 [18]。

D. 实现

Code Critters 是一款 Web 应用程序，设计为可部署在任何服务器上，并可通过互联网访问，使所有人都能在网页浏览器中畅玩。与 Scratch 及许多其他基于积木的编程环境一样，我们决定使用 Blockly[1] 库来构建*Code Critters*，以实现代码的表示和编辑。Scratch 通过定义不同的积木块，让学习者能够将它们组合成有意义的程序，从而在基础编程概念之上增加了一个抽象层。在*Code Critters* 中，我们沿用了这些概念，通过定义不同的积木块来表示被测代码（CUT），而不是使用源代码（图 2 ）。

IV. 结论

Code Critters 是一种概念验证实现，旨在展示如何通过基于积木的编程，以有趣的方式教授测试概念。*CodeCritters* 目前仍在开发中，我们计划在未来为其扩展更多游戏元素、编程概念以及相应的关卡。此外，使用实际学习者对*CodeCritters* 进行研究和评估也十分重要。源代码可在以下地址获取：https://github.com/se2p/code-critters，也可以通过以下网址在线体验*CodeCritters*： https://code-critters.org

致谢

本研究得到德国研究联合会（DFG）的资助，资助编号为 FR 2955/2-1，项目名称为“QuestWare：游戏化的软件测试探索”。

参考资料

[1] Y. B. Kafai 和 Q. Burke，《*Connected code*：为什么儿童需要学习编程》。麻省理工学院出版社，2014年。[2] “2022 年软件工程师现状”，https://hired.com/ 2022-state-of-software-engineers/，访问日期：2023-01-20。[3] F. P. Seth、O. Taipale 和 K. Smolander，“软件测试中的组织与客户相关挑战：对11家软件公司的实证研究”，载于《2014 年 IEEE 第八届信息科学研究挑战国际会议（RCIS）》。IEEE，2014年，第1–12页。[4] D. A. Carrington，“软件测试教学”，载于《ACM SIGCSE 第二届澳大拉西亚计算机科学教育会议论文集，ACSE 1997，墨尔本，维多利亚州，澳大利亚，1997》，ACM 国际会议论文集系列，H. Søndergaard 和 A. J. Hurst 编，第2卷。ACM，1997年，第59–64页。 [在线]。可访问：https://doi.org/10.1145/299359.299369

[5] E. L. Jones，“通过体验式方法将软件测试纳入计算机科学课程体系”，载于《第31届年度教育前沿会议：工程与科学教育的影响。会议论文集（目录号01CH37193）》，第2卷。IEEE，2001年，第F3D–7页。[6] W. Marrero 和 A. Settle，“测试优先：在早期编程课程中强调测试”，载于《第10届年度SIGCSE计算机科学教育创新与技术会议论文集，ITiCSE 2005，葡萄牙卡帕里卡，2005年6月27日至29日》，J. C. Cunha、W. M. Fleischman、V. K. Proulx 和 J. Lourenc,o 编。ACM，2005年，第4–8页。 [在线]。可访问：https://doi.org/10 1145 1067445 1067451 。/。V. Garousi、A. Rainer、P. L. Jr. 和 A. Arcuri，“软件测试教育：系统性文献映射”，《系统与软件杂志》，第165卷，文章编号110570，2020年。 [在线]。可访问：https://doi.org/10.1016/j.jss.2020。110570[8]P. Straubinger 和 G. Fraser，“Gamekins：在Jenkins中将软件测试游戏化”，载于《第44届IEEE/ACM国际软件工程会议：会议伴随论文集，ICSE Companion 2022，美国宾夕法尼亚州匹兹堡，2022年5月22日至24日》。ACM/IEEE，2022年，第85 89 [ ] 10 1145 3510454 3516862 页–。在线。可访问：https://doi.org/。/。S. Deterding、D. Dixon、R. Khaled 和 L. E. Nacke，“从游戏设计元素到游戏性：定义‘游戏化’”，载于《第15届国际学术MindTrek会议：展望未来媒体环境论文集，MindTrek 2011，芬兰坦佩雷，2011年9月28日至30日》，A. Lugmayr、H. Franssila、C. Safran 和 I. Hammouda 编。ACM，2011年，第9–15页。 [在线]。可访问：https://doi.org/10 1145 2181037 2181040 。/。E. M. Raybourn 和 N. Bos，“严肃游戏的设计与评估挑战”，载于《2005年计算机系统中的人为因素会议扩展摘要论文集，CHI 2005，美国俄勒冈州波特兰，2005年4月2日至7日》，G. C. van der Veer 和 C. Gale 编。ACM，2005年，第2049–2050页。 [在线]。可访问：https://doi.org/10 1145 1056808 1057094 。/。M. A. Miljanovic 和 J. S. Bradbury，“Robobug：一款用于学习调试技术的严肃游戏”，载于《2017年ACM国际计算教育研究会议论文集，ICER 2017，美国华盛顿州塔科马，2017年8月18日至20日》，J. Tenenberg、D. Chinn、J. Sheard 和 L. Malmi 编。ACM，2017年，第93–100页。 [在线]。可访问：https://doi.org/10 1145 3105726 3106173 。/。I. S. W. B. Prasetya、C. Q. H. D. Leek、O. Melkonian、J. ten Tusscher、J. van Bergen、J. M. Everink、T. van der Klis、R. Meijerink、R. Oosenbrug、J. J. Oostveen、T. van den Pol 和 W. M. van Zon，“寓学于乐：学习形式化规格”，载于《第41届国际软件工程会议：软件工程教育与培训，ICSE (SEET) 2019，加拿大魁北克省蒙特利尔，2019年5月25日至31日》，S. Beecham 和 D. E. Damian 编。IEEE / ACM，2019年，第192–196页。 [在线]。可访问：https://doi.org/10 1109 2019 00028 。/ICSE-SEET。/。W. Toledo、S. J. Louis 和 S. Sengupta，“Netdefense：一款面向初高中学生的塔防网络安全游戏”，载于《IEEE教育前沿会议，FIE 2022，瑞典乌普萨拉，2022年10月8日至11日》。IEEE，2022年，第1–6页。 [在线]。可访问：https://doi.org/10 1109 56618 2022 9962410 。/FIE。/。G. Fraser、A. Gambi、M. Kreis 和 J. M. Rojas，“用Code Defenders将软件测试课程游戏化”，载于《第50届ACM计算机科学教育技术研讨会论文集，SIGCSE *2019, 27 02,* 美国明尼苏达州明尼阿波利斯，2月至3月》，E. K. Hawthorne、M. A. Perez-Quinones、S. Heckman 和 J. Zhang 编。ACM，2019年，第571–577页。 [在线]。可访问：https://doi.org/10 1145 3287324 3287471 。/。Y. Jia 和 M. Harman，“变异测试发展的分析与综述”，《IEEE软件工程汇刊》，第37卷，第5期，第649–678页，2011年。[16] J. Maloney、M. Resnick、N. Rusk、B. Silverman 和 E. Eastmond，“Scratch编程语言与环境”，《ACM计算教育汇刊》（TOCE），第10卷，第4期，第1–15页，2010年。[17] D. Bau、J. Gray、C. Kelleher、J. Sheldon 和 F. A. Turbak，“可学习编程：积木块及更多”，《ACM通讯》，第60卷，第6期，第72 80, 2017 [ ] 10 1145 3015455 页–。在线。可访问：https://doi.org/。/ J.-F. Collard 和 I. Burnstein，《实用软件测试》，。2002