Inspect4py:一个针对Python代码的知识提取框架储存库

Rosa Filgueira 圣安
德鲁斯大学 英国圣安德
鲁斯，rf208@st-andrews.ac.uk

丹尼尔-加里霍
马德里理工大学（Universidad Politécnica de Madrid
西班牙，马德里
daniel.garijo@upm.es

ABSTRACT

这项工作提出了 **inspect4py**，一个静态代码分析框架设计用于自动提取主要特征、元数据和Python代码存储库的文档。给定一个输入文件夹与代码， inspect4py使用抽象的语法树和状态的艺术工具，以找到所有的函数、类、测试、文档、调用所有代码文件中的图形、模块依赖性和控制流在该存储库中。利用这些发现， inspect4py推断出不同的我们已经评估了调用软件组件的各种方式。我们已经评估了我们的框架在95个有注释的资源库上，获得了有希望的结果。软件类型分类的结果（超过95%的F1分数）。随着inspect4py，我们的目标是简化理解和采用其他研究人员和开发者的软件库。

代码: https://github.com/SoftwareUnderstanding/inspect4py
DOI: https://doi.org/10.5281/zenodo.5907936
许可。开放（BSD3-条款）。

CCS的概念

• **一般和参考** → **调查和概述**；• **应用计算** → **文件采集**；**文件分析**。

关键字

代码挖掘、软件代码、软件分类、软件文档记忆，代码理解

ACM参考格式。
罗莎-菲尔盖拉和丹尼尔-加里霍。2022.Inspect4py:一个知识提取Python代码存储库的框架。In *Proceedings of MSR '22: Pro-第19届国际采矿软件库会议论文集 (MSR 2022)*。ACM，美国纽约，NY，5页。https://doi.org/XXXXXXX。XXXXXXX

1 简介

越来越多的研究结果依赖于软件，在软件本身已经成为一个研究课题，有多个研究诸如使用高效的代码表示法[14]，以应对挑战。函数相似性[7]或从代码中生成文档[4]。

第二，软件对科学的重要性促进了在全世界范围内采用 "可查找、可访问、可互操作和可持续发展 "原则。可重复使用的原则（FAIR）[17]适应于软件[1]。然而。到目前为止，软件重用仍然是一项耗时的任务[11]。

这项工作提出了**expect4py**，一个Python静态代码分析工具。框架，旨在1）提取最相关的信息从一个软件库中包含的所有文件（如函数。类、方法、文档、依赖性、调用图和控制流图），和2）使用提取的信息来分类代码库的类型（即，包、库、脚本或服务）。检测替代的运行方式。

我们的框架旨在促进创建不同的代码特征表示；以及便于代码库的理解[16]。

我们已经用一个带有注释的语料库验证了我们的方法，这个语料库来自于95个Python资源库，显示出令人鼓舞的结果（超过95%）。F1-score）。所有提取的结果都以JSON和HTML形式存储在本地。格式，使其很容易以人类可读的和可接受的方式消费它们。机可读的方式。

从高能物理学[15]到计算物理学[16]等领域。

生物学[12]。研究软件被用来清理和分析数据。

模拟真实系统或可视化科学结果[3]。

在过去的几年里，研究软件已经成为了人们关注的一个话题。

科学界感兴趣的原因主要有两个。首先。

MSR 2022，2022年5月23-24日，美国宾夕法尼亚州匹兹堡。

xxx-xxxx-x/YY/MM..... $15.00
https://doi.org/XXXXXXX.XXXXXXX

本文的其余部分结构如下。第2节介绍了该框架的概况，第3节描述了我们的评 估 。第4节描述了相关的工作和工具，第5节展示了执行的例子，第6节是本文的结论。

2 检验4PY

Inspect4py是一个独立的Python 3软件包，它的开发是为了方便软件的采用和分析。给定一个Python代码库，Inspect4py提取相关的代码特征，检测类和函数元数据，并确定使用它的主要入口点。本节概述了我们框架的主要特征，并对其原理和设计进行了深入探讨。

2.1 概述

Inspect4py将一个代码库文件夹作为输入，并提取两种主要类型的特征。

软件的理解功能，旨在缓解采用...
编写软件包的工作。

- **类和函数元数据和文档**：对于代码库中的每个类，inspect4py 将检索它们的名称、继承类、文档和re-spective方法。对于每个函数，我们的框架会检测其参数、文档、返回值、存储其他函数调用的相关变量以及该函数是否被嵌套（即它在内部定义了更多的函数）。

- **要求**：运行目标软件所需的软件包列表，以及它们的相应版本。
- **测试**：用于测试软件组件功能的文件列表（通常与使用它无关）。
- **软件调用**：一个带有运行目标软件组件的不同备选方案的排名列表，按相关性排序。
- **主要软件类型**：估计目标软件是否旨在成为一个包、库、服务或系列。
  的脚本。

代码特征，旨在从不同的角度来描述代码的特征。

- **文件元数据**：对于每个文件，我们跟踪其包含的类和方法，以及它的依赖关系（导入的模块），是否有声明的主方法，或者文件是否有*主体*（即没有*主*函数的文件，但有对函数/方法的调用和/或实例化类）。
- **控制流图**：对于每个文件，我们检索其控制流图。流程表示为文本文件和图（png、dot或pdf）。控制流图包含了对执行程序的可能路径的备选见解。
- **呼叫图**：对于每个函数（或代码体），我们提取一个所有涉及的函数的调用图，包括那些通过参数、通过赋值或通过动态手段传递的函数。
- **文件层次**：我们记录不同的文件如何被在一个软件库中分组和组织。

因此， inspect4py 会产生一个文件夹，其中包含一个总结性的 JSON 文件（*directory_info.json*），其中有用户选择的特性。还为原始仓库中的每个代码文件创建了一个子文件夹，包括单个层面的JSON结果和控制流结果。下面几节详细说明了这些功能是如何实现的。

2.2 提取类和函数元数据

按照静态代码分析的常见做法[10]， inspect4py使用抽象语法树（Abstract Syntax Trees）[9]来获得完整的代表。

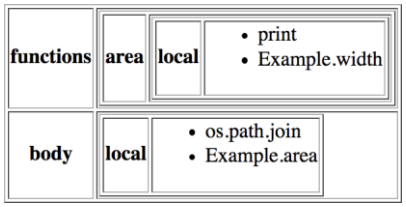

图1: *Example.py*的调用图（清单1）。

2.3 呼叫图和控制流

inspect4py使用解析后的AST，为每个代码文件中的每个函数、*方法*或*主体*创建一个调用图。调用图包括在该函数、方法或*主体*中调用的所有函数。我们支持四种主要的Python函数类型。

- **直接函数**，在代码中直接以其名称提及。
- **参数函数**，作为另一个函数的论据/参数传递。
- **赋值函数**，分配给一个变量，然后用它来进行函数调用。
- **动态函数**，即只在运行时由其输入参数的另一个函数调用决定。

图1是我们结果的一个例子，它描述了解析清单1中的Python脚本后得到的JSON调用图的HTML表示。

每个文件的控制流表示是通过cdmcfparser[2]和staticfg[3]工具实现的，它们以片段的形式创建了代码文件的分层表示。

2.4 模块要求和依赖性

了解一个软件组件的要求和依赖性，对于简化其安装和评估任何安全问题至关重要。Inspect4py在两个层次的粒度上捕获这一信息。

- **文件层面**，通过检查AST中哪些模块是
  在代码库的每个文件中导入。
- **存储库级别**。我们已将Pigar,[4] 一个开放的

对软件库中的所有代码进行分类。特别是，我们使用ast.walk()[1]函数，它可以递归地得到代码树中的所有子节点。

Inspect4py 使用不同的 *ast* 类（如 *FunctionDef*、*Call*、*Assign*、*Return* 等）来自动提取类、方法、函数的所有细节以及它们各自的文档。这种方法允许在内存中捕获树的相关节点，以便进行更多的操作和具体分析，例如，在检测像*main*这样复杂的函数名称时，浏览导入的依赖关系或评估一个函数是否是一个测试。

清单1：Python脚本样本（Example.py）。

Python软件包旨在生成requir

支持Jupyter笔记本。我们选择Pigar是因为它能够处理不同的Python版本（2.7+和3.2+），检索已安装软件包的版本，以及它对Jupyter笔记本的支持。在内部，Pigar也使用*AST*而不是正则表达式，并且能够从*exec*或*eval*表达式、参数和文档字符串中提取导入语句。

2.5 主要软件类型和调用

知道如何轻松地调用代码库对其采用至关重要。这些信息通常可以在README文件中找到，但较少以机器可读的格式出现。我们Inspect4py旨在通过自动检测目标代码库的软件类型（包、库、服务或脚本）来解决这个问题；并找到执行它的其他方法的清单。

[2] https://github.com/SergeySatskiy/cdm-flowparser

```
输入os
path = os.path.join("/User","/home","file.txt")
def width():
    返回 5
def area(length,func)。
    print(length*func())
area(5,width)
```

[1] https://docs.python.org/3/library/ast.html?highlight=walk#ast.walk

[3] https://github.com/coetaur0/staticfg

[4] https://github.com/damnever/pigar

*2.5.1 软件类型。*我们区分了四种主要的软件类型。

- **包**：旨在通过命令行执行的工具。
- **库**：旨在作为其他软件的一部分被导入的代码。
- **服务**：主要功能是启动一个网络服务的代码，通常是通过一个脚本。
- **脚本**：不符合前面任何类别的代码，在一个或几个可执行文件中运行（通过一个主函数，或一个带*主体*的脚本）。

虽然许多 Python 代码库可能只被归入其中一类，但发现许多代码库有重叠的情况也很常见。例如，一个包可能作为一个库被导入，或者一个库也可能有一个演示脚本来详细说明如何使用它。因此， inspect4py 返回目标软件库的*主要*估计功能，以及运行它的备选方案的排序列表。

*2.5.2 软件调用和类型分类。*我们根据打包Python项目的最佳实践[5]和对目标库中所有代码文件的分析，创建了一个启发式方法。我们的重点是

文件的AST来识别每个文件中的*主要*功能，以及那些没有主要功能，但有*主体*的可执行文件。在这两种情况下，我们将文件标记为**脚本**。

我们在具有主要功能的脚本中进行额外的分析，以提供更多的洞察力：我们使用深度优先搜索来探索它们的调用图，并确定它们与其他具有主要功能的脚本的依赖关系（直接或间接）。如果一个脚本被另一个脚本导入，它将被标记为*二级脚本*。对于每个脚本，我们的结果也会存储其全部的导入脚本列表。

*2.5.3 对软件调用方法进行排名。*一旦我们的文件分析完成，我们就通过估计它们的价 值 对 我们的调用结果进行排名。我们有一个评分函数，它根据之前分析得出的特征给每个结果分配权重。根据作者在90多个代码库中观察到的经验性做法，不同的特征有不同的权重。打分函数定义为：。

$$score(x) = {}_{wl}ib * lib(x) + {}_{wreadme} * readme\ (x) + \\ + {}_{wser\ vice} * service\ (x) + \\ + {}_{wm}ain * main(x) + {}_{wbod\ y} * body\ (x) \quad (1)$$

元数据文件，包括动态的（*setup.py*）和静态的（*setup.cfg*），*有主*函数的文件，以及有*主体*的文件。

我们的启发式方法首先寻找版本库是否有元数据文件（'setup.py'或'setup.cfg'文件（或两者）），因为这些文件通常包含关于代码库自身的有用信息。如果找到了，我们就寻找。1）软件项目名称；2）入口点，通常是控制台脚本、函数或其他由开发者确定的可调用的类似函数的对象，可以作为命令行工具使用。

如果发现了入口点，我们就检查是否有控制台脚本与软件组件的名称有明显的重叠。这样做的理由是，在大多数观察到的情况下，软件包名称和控制台入口点是非常相似的。如果有明显的重叠，我们就把软件库标记为**一个包**。如果没有，或者没有入口点，我们就认为该代码库是一个**库**。接下来，我们对软件库的其他文件进行单独分析。

服务文件。使用存储在AST中的信息，我们检查如果分析的文件中导入的模块和库与创建Web服务常用的任何主要库相对应。[6] 如果是，我们就把该文件标记为**服务**脚本。

测试文件。许多软件项目包括测试文件，以确保新的更新不会改变所开发的应用程序的预期行为。这些文件在应用程序开发中是至关重要的，但对于理解和采用一个软件组件却不那么重要。我们通过审查是否包括测试框架来检测测试在文件依赖关系中，[7]并通过评估大多数功能是否

文件中使用*assert*函数，在测试应用程序时通常使用。如果一

个测试文件有一个*主*函数，我们也会在结果中对其进行注释。**脚本文件**。如果一个文件没有被标记为测试或服务，我们会检查无论它是否是一个可执行文件。如上所述，我们使用

其中*x*表示被评分的调用，*lib(x)*=1，如果代码库被确定为库或包（否则为零），*service(x)*=1，如果x被确定为服务，*main(x)*和*body(x)*等于1，如果x有一个主函数或只有主体（否则它们等于零）。我们还考虑了一个服务或脚本是否在README文件中被提及（*readme(x)* = 1），因为这表明该服务或脚本被软件作者确定为重要的。与这些特征中的每一个相关的权重（用*w*表示。如wli_b），对于包或库来说是最高的，其次是服务，最后是脚本。如果一个可执行文件有一个*主*功能，它的权重将高于只有*主体*的脚本。

清单2：显示两种调用方式的JSON片段

```
"software_invocation":[]。
    {
      "运行":[[
           "幽门"。
           "pylode.cli:main,"
        ]
       ],
      "类型":"包"。
      "installation":"pipinstallpyLODE","r
      anking":1
    },
    {
      "类型":"Service"。
      "run":"python pyLODE/pylode/server.py",
      "has_structure":"body"。
      "mentioned_in_readme":true,
      "ranking":2
    }
  ],
  "software_type":"包"
```

一旦所有的调用替代方案都被评估了，我们就按分数从高到低进行排序。为了提高可读性，我们建立了一个升序的排名，其中第一个位置是

被分配给得分最高的条目。排名第一的

[5] https://packaging.python.org/en/latest/tutorials/packaging-projects/
[6]个服务列表：flask, flask_restful, falcon, falcon_app, aiohttp, bottle, django, fastapi, locust, pyramid, hug, eve, connexion
[7]个测试库列表：unittest, pytest, nose, nose2, doctest, testify, behave, lettuce

元素被返回作为主要的软件类型。如果两个调用备选方案具有相同的分数，它们就会被分配相同的排名号。清单2显示了一个简单的包的例子，其中的

主要的调用是通过命令行，但这也有
作为一种服务的替代性调用。Inspect4py检测到的主要
命令来运行软件包，（可在*setup.cfg*中找到）。
代码库），以及另一个脚本（*server.py*）。
在主README中提到。

3 初步评估

对 inspect4py 的初步评估将软件类型和调用结果与人工注释的语料库进行比较。

3.1 有注释的语料库

我们已经创建了两个不同的语料库来评估我们的方法。对于主要的软件类型检测，我们选择了一个由95个不同的Python代码库组成的语料库（分布在24个包、27个库、13个服务和31个脚本中）。[8] 所有的库都由每个作者根据其文档分别进行了注释，并进行了比较，直到达成一致。这些资源库的范围和领域各不相同，从用于机器学习的研究资源库[9]到流行的社区工具，如Apache Airflow[10]或特定领域的库（如Astropy[11]）。为了获得更广泛的代表性样本，我们还包括了作者所在机构开发的库，这些库的可用文档较少或正在开发中。

第二个语料库是用来评估排名结果的，它是作为第一个语料库的一个子集产生的。对于所有具有多种调用方法的资源库，作者根据资源库的结构和文档，手动注释了最相关的文件。结果，有44个资源库被注释了。

3.2 结果

表1显示了我们对主要软件类型分类的结果概览，每个支持的类别都是如此。我们的启发式方法，基于Python应用程序开发的最佳实践，在所有类别中产生了超过95%的F1分数。我们遇到的唯一错误发生在开发者超出常规做法的时候，比如为他们的库创建自定义元数据文件。

为了评估我们的排名结果，我们选择了归一化折扣累积收益（NDGC）[5]，这是一个用于信息检索的指标，用来评估排名质量。NDGC的范围从0（最小）到1（最大）。我们的综合排名为0.87，对于初步评估来说，这被认为是令人满意的。

4 相关的工作

许多静态代码分析工具已经被开发出来，用于提取代码元数据、文档或特征[10]。从机器学习的角度来看，[14]描述了一项关于代码特征提取技术的调查，以训练源代码模型。像libsa4py[8]这样的工具从代码中创建特征，以补充 inspect4py中已经提取的信息。

软件类型	精度	召回	F1分值
包装	1	0.916	0.956
图书馆	0.93	1	0.9637
服务	1	1	1
脚本	0.967	0.967	0.967

表1:软件类型分类的结果。

其他工具与我们框架中的功能有一些重叠。例如，pydeps[12]和modulegraph2[13]对模块的依赖性进行分析，像*pydoc*这样的库[14]，生成HTML代码文档，像*Pigar*这样的库[15]，提取代码需求，像*pyan*[2]和*pycg*[13]这样的包使用静态分析为Python代码生成调用图，包括高阶函数、类继承方案和嵌套函数定义。Inspect4py建立在这些工具（例如pygar）的基础上，将它们的所有功能整合到一个单一的框架中，并对它们的所有结果使用一个独特的表示法。

据我们所知，目前还没有检测软件类型和调用方法的框架。

5 安装和执行

Inspect4py可以通过pip（`pip install inspect4py`）和Docker来安装。要调用该工具的基本功能（即提取类、函数及其文档），需要运行以下命令。

```
inspect4py -i <inputfile.py|directory>
```

这个命令可以用不同的选项来限定，[16] 以便执行不同的功能。例如，下面的命令提取了类、函数和方法的文档，同时也存储了软件的调用信息（标志-*si*）。

```
inpsect4py -i repository_path -o out_path -si -html
```

6 结论和未来工作

本文介绍了 inspect4py，这是一个静态代码分析框架，旨在从Python代码库中提取常见的代码特征和文档，以便于理解。初步评估显示，评估Python代码库类型的结果很有希望，还可以识别和排列运行这些代码库的其他方式（从而为潜在用户节省时间）。

我们正在将 inspect4py 与软件元数据提取工具[6]结合使用，为完成 README 文件提出建议。
在寻找类似的代码时，比较软件的其他表现形式。

至于未来的工作，我们正在扩展 inspect4py，以便 1）通过注释更多数量的资源库来改进我们的评估结果，2）纳入新的特征提取工具（例如 libsa4py），以及 3）改进我们的调用检测以包括说明性的参数实例。

[12] https://github.com/thebjorn/pydeps

[13] https://pypi.org/project/modulegraph2/

[8] 基准可

在以下网站获得：https://doi.org/10.5281/zenodo.5907936

[9] https://papers with code.com/

[10] https://github.com/apache/airflow

[11] https://github.com/astropy/astropy

[14] https://docs.python.org/3/library/pydoc.html

[15] https://github.com/damnever/pigar

[16] 见 https://github.com/SoftwareUnderstanding/inspect4py/blob/main/README.md

[17] https://github.com/SoftwareUnderstanding/completeR

参考文献

[1] Neil P. Chue Hong, Daniel S. Katz, Michelle Barker, Anna-Lena Lamprecht, Carlos Martinez, Fotis E. Psomopoulos, Jen Harrow, Leyla Jael Castro, Morane Gruen- peter, Paula Andrea Martinez, Tom Honeyman, Alexander Struck, Allen Lee, Axel Loewe, Ben van Werkhoven, Catherine Jones, Daniel Garijo, Esther Plomp,Fran- coise Genova, Hugh Shanahan, Joanna Leng, Maggie Hellström, Malin Sandström, Manodeep Sinha, Mateusz Kuzak, Patricia Herterich, Qian Zhang, Sharif Islam, Susanna-Assunta Sansone, Tom Pollard, Dwi Atmojo;Udayanto, Alan Williams, Andreas Czerniak, Anna Niehues, Anne Claire Fouilloux, Bala Desinghu, Carole Goble, Céline Richard, Charles Gray, Chris Erdmann, Daniel Nüst, Daniele Tar- tarini, Elena Ranguelova。 Hartwig Anzt, Ilian Todorov, James McNally, Javier Moldon, Jessica Burnett, Julian Garrido-Sánchez, Khalid Belhajjame, Laurents Sesink, Lorraine Hwang, Marcos Roberto Tovani-Palone, Mark D. Wilkinson, Mathieu Servillat, Matthias Liffers, Merc Fox, Nadica Miljković, Nick Lynch, Paula Martinez Lavanchy, Sandra Gesing, Sarah Stevens, Sergio Martinez Cuesta, Silvio Peroni, Stian Soiland-Reyes, Tom Bakker, Tovo Rabemanantsoa, Vanessa Sochat, and Yo Yehudi。 2021. 研究软件的公平原则（FAIR4RS 原则）。 (2021)。 https://doi.org/10.15497/RDA00065 出版商。研究数据 联盟。

[2] D.Fraser, E. Horner, J. Jeronen, and P. Massot.2020.Pyan3:Python 3的离线调用图生成器。 https://github.com/davidfraser/pyan 。 Accessed:09-January-2022.

[3] Morane Gruenpeter, Daniel S. Katz, Anna-Lena Lamprecht, Tom Honeyman, Daniel Garijo, Alexander Struck, Anna Niehues, Paula Andrea Martinez, Leyla Jael Castro, Tovo Rabemanantsoa, Neil P. Chue Hong, Carlos Martinez- Ortiz, Laurents Sesink, Matthias Liffers, Anne Claire Fouilloux, Chris Erd- mann, Silvio Peroni, Paula Martinez Lavanchy, Ilian Todorov 和 Manodeep Sinha。 2021.定义研究软件：一个有争议的讨论。 https: //doi.org/10.5281/zenodo.5504016

[4] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin.2018.深度代码注释生成。In *Proceedings of the 26th Conference on Program Comprehension* (Gothenburg, Sweden) *(ICPC '18)*.Association for Computing Machinery, New York, NY, USA, 200-210. https://doi.org/10.1145/3196321.3196334

[5] Kalervo Järvelin and Jaana Kekäläinen.2002.Cumulated Gain-Based Evaluation of IR Techniques.*ACM Trans.Inf.Syst.*20, 4 (oct 2002), 422-446. https://doi.org/10.1145/582415.582418

[6] 艾丹-凯利和丹尼尔-加里霍。 2021.创建科学软件元数据知识图谱的框架.*定量科学研究*（11 2021），1-37. https://doi.org/10.1162/qss_a_00167 arXiv:https://direct.mit.edu/qss/article-pdf/doi/10.1162/qss_a_00167/1971225/qss_a_00167.pdf

[7] Sifei Luan, Di Yang, Celeste Barnaby, Koushik Sen, and Satish Chandra.2019.Aroma: Code Recommendation via Structural Code Search.*Proc.ACM Program.Lang.*3, OOPSLA, Article 152 (oct 2019), 28 pages. https://doi.org/10.1145/3360578

[8] Amir M. Mir, Evaldas Latoškinas, and Georgios Gousios.2021.ManyTypes4Py: 基于机器学习的类型推理的基准Python数据集。在*2021年IEEE/ACM第18届采矿软件库国际会议上（MSR）*。 585–589. https://doi.org/10.1109/MSR52588.2021.00079

[9] Iulian Neamtiu, Jeffrey S. Foster, and Michael Hicks.2005.Using Abstract Syntax Tree Matching Understanding Source Code Evolution.*SIGSOFT Softw.工程。Notes* 30, 4 (May 2005), 1-5. https://doi.org/10.1145/1082983.1083143

[10] Jernej Novak, Andrej Krajnc, and Rok Žontar.2010.静态代码的分类 分析工具。在*第33届国际会议MIPRO*。418-422.

[11] 杰弗里-M-佩克尔。 2020.对科学家的挑战：你十年前的代码还能 运行吗？*自然》*584，7822（2020），656-659。

[12] Andreas Prlić 和 Hilmar Lapp.2012.PLOS计算生物学软件 部分。*PLoS计算生物学*8, 11 (2012), e1002799。

[13] Vitalis Salis, Thodoris Sotiropoulos, Panos Louridas, Diomidis Spinellis, and Dimitris Mitropoulos.2021.PyCG: Practical Call Graph Generation in Python.在*2021年IEEE/ACM第43届国际软件工程会议（ICSE）上*。 1646–1657. https://doi.org/10.1109/ICSE43902.2021.00146

[14] Tushar Sharma, Maria Kechagia, Stefanos Georgiou, Rohit Tiwari, and Federica Sarro.2021.关于源代码分析的机器学习技术的调查。 arXiv:2110.09610 [cs.SE]

[15] HEP软件基金会, Johannes Albrecht, Antonio Augusto Alves, Guilherme Amadio, Giuseppe Andronico, Nguyen Anh-Ky, Laurent Aphe- cetche, John Apostolakis, Makoto Asai, Luca Atzori, Marian Babik, Giuseppe Bagliesi, Marilena Bandieramonte, Sunanda Banerjee, Martin Barisits, Lothar A.T.Bauerdick, Stefano Belforte, Douglas Benjamin, Catrin Bernius, Wahid Bhimji, Riccardo Maria Bianchi, Ian Bird, Catherine Biscarat, Jakob Blomer, Kenneth Bloom, Tommaso Boccali, Brian Bockelman, Tomasz Bold, Daniele Bonacorsi, Antonio Boveia, Concezio Bozzi, Marko Bracko, David Britton, Andy Buckley,Predrag Buncic, Paolo Calafiura, Simone Campana, Philippe Canal, Luca Canali, Gianpaolo Carlino, Nuno Castro, Marco Cattaneo, Gianluca Cerminara, Javier Cervantes Villanueva, Philip Chang, John Chapman, Gang Chen, Tay- lor Childers, Peter Clarke, Marco Clemencic, Eric Cogneras, Jeremy Coles, Ian Collier, David Colling, Gloria Corti, Gabriele Cosmo, David de Costanzo, Ben Cou

-turier, Kyle Cranmer, Jack Cranshaw, Leonardo Cristella, David Crooks, Sabine Crépé-Renaudin, Robert Currie, Sünje Dallmeier-Tiessen, Kaushik De, Michel De Cian, Albert De Roeck。Antonio Delgado Peris, Frédéric Derue, Alessandro Di Girolamo, Salvatore Di Guida, Gancho Dimitrov, Caterina Doglioni, Andrea Dotti, Dirk Duellmann, Laurent Duflot, Dave Dykstra, Katarzyna Dziedziniewicz- Wojcik, Agnieszka Dziurda, Ulrik Egede, Peter Elmer, Johannes Elmsheuser, V.Daniel Elvira, Giulio Eulisse, Steven Farrell, Torben Ferber, Andrej Filipcic, Ian Fisk, Conor Fitzpatrick, José Flix, Andrea Formica, Alessandra Forti, Giovanni Franzoni, James Frost, Stu Fuess, Frank Gaede, Gerardo Ganis,Robert Gardner, Vincent Garonne, Andreas Gellrich, Krzysztof Genser, Simon George, Frank Geurts, Andrei Gheata, Mihaela Gheata, Francesco Giacomini, Stefano Giagu, Manuel Giffels, Douglas Gingrich, Maria Girone, Vladimir V。Gligorov, Ivan Glushkov, Wesley Gohn, Jose Benito Gonzalez Lopez, Isidro González Caballero, Juan R。González Fernández, Giacomo Govi, Claudio Grandi, Hadrien Grasland, Heather Gray, Lucia Grillo, Wen Guan, Oliver Gutsche, Vardan Gyurjyan, An- drew Hanushevsky, Farah Hariri, Thomas Hartmann, John Harvey, Thomas Hauth, Benedikt Hegner, Beate Heinemann, Lukas Heinrich, Andreas Heiss, José M。Hernández, Michael Hildreth, Mark Hodgkinson, Stefan Hoeche, Burt Holzman, Peter Hristov, Xingtao Huang, Vladimir N。Ivanchenko, Todor Ivanov, Jan Iven, Brij Jashal, Bodhitha Jayatilaka, Roger Jones, Michel Jouvin, Soon Yung Jun, Michael Kagan, Charles William Kalderon, Meghan Kane, Edward Karavakis, Daniel S。Katz, Dorian Kcira, Oliver Keeble, Borut Paul Kersevan, Michael Kirby, Alexei Klimentov, Markus Klute, Ilya Komarov, Dmitri Konstantinov, Patrick Koppenburg, Jim Kowalkowski, Luke Kreczko, Thomas Kuhr, Robert Kutschke, Valentin Kuznetsov, Walter Lampl。Eric Lancon, David Lange, Mario Lassnig, Paul Laycock, Charles Leggett, James Letts, Birgit Lewendel, Teng Li, Guilherme Lima, Jacob Linacre, Tomas Linden, Miron Livny, Giuseppe Lo Presti, Sebas- tian Lopienski, Peter Love, Adam Lyon, Nicolò Magini, Zachary L。Marshall, Edoardo Martelli, Stewart Martin-Haugh, Pere Mato, Kajari Mazumdar, Thomas McCauley, Josh McFayden, Shawn McKee, Andrew McNab, Rashid Mehdiyev, Helge Meinhard, Dario Menasce,Patricia Mendez Lorenzo, Alaettin Serhan Mete, Michele Michelotto, Jovan Mitrevski, Lorenzo Moneta, Ben Morgan, Richard Mount, Edward Moyse, Sean Murray, Armin Nairz, Mark S.Neubauer, Andrew Norman, Sérgio Novaes, Mihaly Novak, Arantza Oyanguren, Nurcan Ozturk, Andres Pacheco Pages, Michela Paganini, Jerome Pansanel, Vincent R.帕斯库兹，格伦-帕特里克，亚历克斯-皮尔斯，本-皮尔森，凯文-佩德罗，加布里埃尔-珀杜，安东尼奥-佩雷斯-卡莱罗-伊兹奎尔多，卢卡-佩罗兹，特罗斯-彼得森，马科-佩特里奇，安德烈亚斯-佩佐尔德，约纳坦-皮德拉，里奥-皮洛宁，达尼洛-皮帕罗，吉姆-皮瓦尔斯基。Witold Poko- rski, Francesco Polci, Karolos Potamianos, Fernanda Psihas, Albert Puig Navarro, Günter Quast, Gerhard Raven, Jürgen Reuter, Alberto Ribon, Lorenzo Rinaldi, Martin Ritter, James Robinson, Eduardo Rodrigues,Stefan Roiser, David Rousseau, Gareth Roy, Grigori Rybkine, Andre Sailer, Tai Sakuma, Renato Santana, Andrea Sartirana, Heidi Schellman, Jaroslava Schovancová, Steven Schramm, Markus Schulz, Andrea Sciabà, Sally Seidel,Sezen Sekmen, Cedric Serfon, Horst Severini, Elizabeth Sexton-Kennedy, Michael Seymour, Davide Sgalaberna, Illya Shapo- val, Jamie Shiers, Jing-Ge Shiu, Hannah Short, Gian Piero Siroli, Sam Skipsey, Tim Smith, Scott Snyder, Michael D.Sokoloff, Panagiotis Spentzouris, Hartmut Stadie, Giordon Stark, Gordon Stewart, Graeme A.Stewart, Arturo Sánchez, Alberto Sánchez-Hernández, Anyes Taffard, Umberto Tamponi, Jeff Templon, Gi- acomo Tenaglia, Vakhtang Tsulaia, Christopher Tunnell, Eric Vaandering, Andrea Valassi,Sofia Vallecorsa, Liviu Valsan, Peter Van Gemmeren, Renaud Vernet, Brett Viren, Jean-Roch Vlimant, Christian Voss, Margaret Votava, Carl Vuosalo, Carlos Vázquez Sierra, Romain Wartel, Gordon T。Watts, Torre Wenaus, Sandro Wenzel, Mike Williams, Frank Winklmeier, Christoph Wissing, Frank Wuerthwein, Ben- jamin Wynne, Zhang Xiaomei, Wei Yang, and Efe Yazgan.2019.2020年HEP软件和计算研发路线图》。 *大科学的计算和软件* 3, 1 (Dec. 2019), 7. https://doi.org/10.1007/s41781-018-0018-8

[16] A.Von Mayrhauser和A.M. Vans。1995.软件维护和进化过程中的程序理解。*Computer* 28, 8 (1995), 44-55. https://doi.org/10.1109/2.402076

[17] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J。Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A。C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Al- bert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons.2016.科学数据管理和监管的FAIR指导原则。*科学数据*3（2016年3月），160018。https://doi.org/10.1038/sdata.2016.18