·计算机软件理论、技术与应用·

代码重复检测结果可视化设计与实现

刘 旭

(SAP 中国研究院商务智能部,上海 201203)

摘 要: 文章阐述了代码重复检测的意义及检测技术的基本原理,深入分析了树图(Treemap)和层次边聚合图(Hierarchical Edge Bundles)可视化技术的特性。根据代码重复检测数据结构的特点,文章总结了代码重复检测结果可视化的设计目标,提出使用树图和层次边聚合图作为可视化形式的方案。在此方案中,将树图作为代码层次关系的可视化形式,将层次边聚合图作为代码重复关系的可视化形式。文章进一步分析了系统中的数据交换格式及关键技术,设计并使用多种软件技术实现了代码重复检测结果可视化的原型系统 VizCD。实验结果表明,该可视化系统可用于代码重复检测结果分析中。

关键词: 软件度量; 代码重复; 可视化; 树图; 层次边聚合图

中图分类号: TP391 **文献标志码:** A **文章编号:** 1673 - 159X(2017)06 - 0013 - 10

doi: 10.3969/j.issn.1673 - 159X.2017.06.003

Code Duplication Detection Results Visualization Design and Implementation

LIU Xu

(*Department of Business Intelligence of SAP Labs China, Shanghai* 201203 *China*)

Abstract: This paper discusses the significance of code duplication detection, the basic principles of code duplication detection technology, and the features of the Treemap and Hierarchical Edge Bundles visualization technology. According to the characteristics of code duplication detection data structure, this paper summarizes design goal of code duplication visualization, and then proposes a solution by using Treemap and Hierarchical Edge Bundles as visualization representations. In the solution, Treemap is used as the visualization representation of code hierarchical relationships, and Hierarchical Edge Bundles is used as the visualization representation of code duplication relationships. According to the solution, this paper analyzes data exchange format and key technologies of system implementation, designs and implements code duplication visualization prototype system VizCD with a variety of software technologies, and conducts experiments to prove the role of the visualization system in code duplication detection result analysis.

Keywords: software metric; code duplication; visualization; Treemap; Hierarchical Edge Bundles

代码重复检测又称为代码克隆检测,是软件度量的一个方面。代码重复在软件开发实践中经常发生,一般是指几个非常相似的代码片段,它们虽然仅从代码文本上看不一定一致,但是在程序逻辑上几乎等同[1]。

一般认为,代码重复应该在代码编写过程中尽量避免[2]。当软件规模越来越大时,代码重复的检测结果数据集也变得越来越大,直接查看数据集的效率很低;然而,使用可视化技术可以更方便地分析这些数据并从中找到需要改进的程序模块。

对软件数据的分析贯穿着软件开发、测试、维护等软件生命周期的各个方面。软件可视化作为软件数据分析和展示的手段,在软件工程中的使用越来越广泛。软件可视化实现的关键是针对数据的特点引入合适的可视化形式和交互方式。可视化的难点在于软件数据的抽象性,即软件数据一般没有明显的几何结构,必须为软件数据寻找适当的视觉隐喻。

收稿日期: 2017 - 03 - 03

作者简介: 刘旭(1982—),男,硕士,主要研究方向为商务智能、信息可视化及教育技术。E - mail: liuxuhere@hotmail.com

引用格式: 刘旭. 代码重复检测结果可视化设计与实现[J]. 西华大学学报(自然科学版),2017,36(6):13.

作为软件可视化分析的一部分，代码重复的可视化分析在软件质量保证、软件性能调优、软件重构等方面有重要作用[3]。目前的软件可视化研究中，针对代码重复检测结果的可视化研究还比较少，使用的可视化形式也比较简单，例如散点图(scatter plot)，只能用于分析2个文件之间的代码重复[4]。虽然现在的代码重复检测工具很多，检测能力也越来越强;但是检测结果一般都是以文本方式输出程序日志，或是生成结构化的XML文件，单纯查阅程序日志或是浏览文件很难全面了解重复代码的分布情况[5]。合适的可视化方案可以作为代码重复检测结果分析的有力工具，极大地提高对代码重复分析的效率，从而改善软件测试和重构的质量[6]。

代码重复检测的数据结构，从代码块和代码文件的角度看，是一种层次结构的数据;从重复的代码块之间的关系来看，又是一种网络数据。一个良好的代码重复检测结果可视化形式，应该是既能反映代码的层次关系，又能揭示不同的重复代码块之间的联系，这给信息可视化带来了挑战。本文试图针对代码重复的数据提出一种可视化方案，并设计实现一个可视化原型系统VizCD。

1 代码重复的检测

代码重复的特点在于一个单元可能与多个单元有不同的重复。图1是一个典型的代码重复示意图[7]。可以看到，File1、File2与File3都有相同的重复代码片段，File3与File4有另外的代码重复片段。

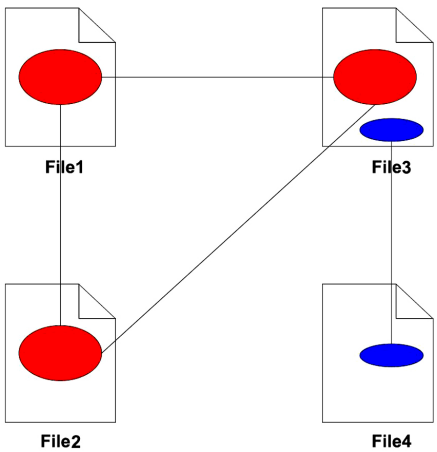


图1 典型的代码重复示意图

代码重复的检测相当困难。代码重复检测程序无法事先知道哪一个代码片段可能被重复。理论上说，每一个代码片段都应该与其他可能的片段比较，这一比较将是非常耗时的过程。代码重复检测的另一个难点是代码重复并非指代码完全一样，而是指2段代码的逻辑功能相似。

代码重复检测的基本原理是通过对程序代码特征进行相似性分析以找出重复的代码片段[8]。检测的核心问题是对代码重复部分的匹配技术。匹配技术的选择将决定代码的中间格式和匹配算法。目前的匹配技术所分析的对象可以大致分为4种类型：文本、词法、语法、语义。文本分析的方法使用的是几乎没有转化过的源代码进行比较。一种典型的文本比较方式是计算一个代码片段的Hash值，然后去检查有相同Hash值的代码片段。滑动窗口技术和增量Hash函数可以用来加速这种检查。词法分析(也称为基于符号的分析)是通过将源代码转换成一个词法符号序列进行的，此过程类似编译器使用的词法分析。语法分析的主要思想是使用语法解析器将源程序转换为抽象语法树(AST)或其他语法结构，然后使用树匹配或结构度量的方式来找到代码重复[9]。语义分析方法使用的也是静态程序分析，但是它试图提供比语法相似性分析更精确的比较结果。此方法一般是将程序表示为程序依赖图(PDG)，以图上的结点代表程序的表达式和语句，图上的边用来表示控制和数据依赖性。

为减少不必要的比较，一般需要在检查之前对代码进行预处理，去除不需要比较的部分(比如LEX或YACC自动生成的代码)，然后将代码转换到某种中间格式，通过匹配检测以确定代码重复对。用于匹配检测的算法可能是简单的文本比较，也可能是其他更复杂的算法，如后缀树匹配和Hash值比较。在比较完成之后，需要将中间格式表示的代码重复片段映射到源代码，然后可以对结果进行人工分析，也可以进行自动的启发式过滤，还可以对代码重复结果进行聚合处理，最终获得的结果即可用于代码重复的可视化分析。目前不少代码检测工具还处在实验阶段，难以用于实用程序[10]。Simian是比较成熟的代码重复检测工具，时间与空间性能都很好[11]。本文的可视化原型系统VizCD将支持Simian作为检测工具。

代码重复的检测可以基于代码结构的各个层次，可能是文件夹、包、类、文件、函数。从可视化的角度来看，可以将这些统称为检测单元。不同的程序设计语言可能有不同的检测单元。常见的代码重复检测单元是文件，一般都使用文件作为组织源代码的方式。文件比较是软件开发领域的一个常

见问题,目前已经有大量成熟的文件比较工具可用,以文件为粒度的重复检测报告可以方便软件测试和开发人员定位及修改文件。

代码重复检测结果中往往会列出包含重复代码的文件路径和文件名,以及重复代码片段的行号或行数。2 个有代码重复的文件被称为重复对,而多个文件可能有共同的重复代码,这些文件的集合被称为重复类,一个重复类里面有一个或多个重复对。图 1 所示的典型的代码重复情况中,File1、File2 与 File3 都有相同的重复代码片段,所以构成一个重复类,这个重复类中的重复对为(File1,File2)、(File2,File3)、(File1,File3)。File3 与 File4 有另外的代码重复片段,构成另一个重复类,这个重复类只有一个重复对(File3,File4)。不同的重复对,重复的代码行数和代码位置可能是不同的。本文对于代码重复检测结果的可视化研究,重点在于由不同文件构成的文件重复对的可视化。

2　基于 Treemap 的代码重复检测结果可视化

软件可视化系统是一类特殊的信息可视化系统。现代信息可视化系统的设计,一般都要求做到直观化、关联化、艺术化、交互性[12],即要求功能性和易用性的统一。具体到代码重复的检测结果可视化,应考虑以下几方面。

1) 显示检测单元的详细信息。检测单元往往具有层次结构,比如 Java 语言实现的软件源代码中,类与它的子类构成了层次关系,在很多 Java 开发环境中,对类信息的可视化反映了这种层次关系。以文件检测单元的情况就更加典型,由于源代码不同的文件放在不同的文件夹,文件与文件夹构成了层次结构。显示文件层次关系的信息有利于文件的定位和浏览,并且方便使用文件比较工具和代码编辑器进行进一步处理[13]。

2) 显示检测单元之间的关系。以文件为例,一个文件可以与多个文件构成重复对,各个文件之间实际上是网络关系。代码重复的可视化需要结合层次数据可视化与网络数据可视化的设计。

3) 能处理较多数据。由于代码文件数量巨大,如果将数据全部显示,可能存在性能问题。海量数据可能超过软硬件的处理能力,不能在用户可接受时间内显示结果,即使显示,也可能由于数据的表示过于密集导致缺乏实用价值。在可视化设计过程中,一般情况下,对大数据量可以考虑将某些数据聚合,并且为可视化形式添加交互能力,利用 zoom in 方式显示用户选取的细节。

4) 可扩展性。代码重复的可视化除了显示重复的代码块之外,一般还需要有选择性地显示其他的软件度量指标,如代码行数、嵌套层次。在代码重复的可视化工具中,应该考虑加入其他指标的可能。

代码文件系统是典型的层次数据,传统的可视化形式是树形图。树形图的特点是用几何形状或文字代表数据结点,用线段表示层次数据之间的关系,这样整个结构看起来就像一棵树; 但是树形图的连接线段之间存在大量空白无法利用,当数据量增加时,子结点会逐渐密集排列在一起,视觉复杂性显著增加,其显示效果令人难以接受。Treemap 使用具有一定面积的块或体来表示数据结点,利用结点之间的位置关系来表示数据之间的层次关系[14]。由于代码文件是典型的层次结构,如果不考虑代码文件之间因代码重复导致的相互关系,仅考虑代码文件中重复行的多少,使用 Treemap 对其进行层次关系的可视化非常方便。

相对于树形图,Treemap 的优点是充分利用空间[15],通过结点的大小、颜色表示数据的各种属性[16]。同时,Treemap 的结点位置还可以表示数据的分布关系。Treemap 的基本形式是用一个矩形区域表示根结点,将根结点划分为多个矩形区域以表示子结点,这样递归地表示整个层次结构数据[17]。图 2 描述了一个 Treemap 的生成步骤[18]。

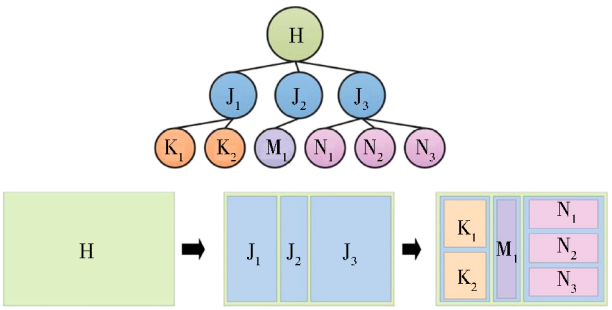


图 2　Treemap 的一般生成步骤

典型的 Treemap 结点一般有大小、颜色、位置 3 个属性,可以表示 3 个维度的数据。代码文件没有地理信息的特点,而且 Treemap 的很多布局算法不能保证矩形区域的位置稳定性; 因此,位置这个属性在代码文件的表示中可以忽略。矩形区域的大小可以用于表示代码文件的行数,颜色可以作为重复度的标志,重复的代码行数越多,颜色越深。需要注意的是,由于一个文件可以属于多个重复对,对文件中代码重复的行数需要进行聚合。表 1 是对图 1

所示的典型情况进行代码重复检测的结果,表中列出了所有的重复对。

表 1 代码重复检测的结果示例

File1	File1 行数	File2	File2 行数	重复行数
folder1\file2. java	464	folder2\file3. java	2 001	458
folder1\file1. java	464	folder2\file3. java	2 001	458
folder1\file1. java	464	folder1\file2. java	464	458
folder2\file4. java	1 540	folder2\file3. java	2 001	1 533

代码重复检测结果包含了 2 个重复类,共计 4 个重复对。可以看出,file1. java 与 file2. java、file3. java 都有代码重复,file3. java 与 3 个文件都有重复。在可视化重复数据的时候,应该把每个文件中的重复代码行数进行累加。累加结果为:

file1. java: 458 + 458 = 916。

file2. java: 458 + 458 = 916。

file3. java: 458 + 458 + 1533 = 2449。

file4. java: 1533。

图 3 示出使用 Treemap 对此代码重复检测结果可视化的例子。可以看出: file3. java 文件行数最多,因此占有最大的面积; 代码重复行数也最多,因此颜色是最深的。

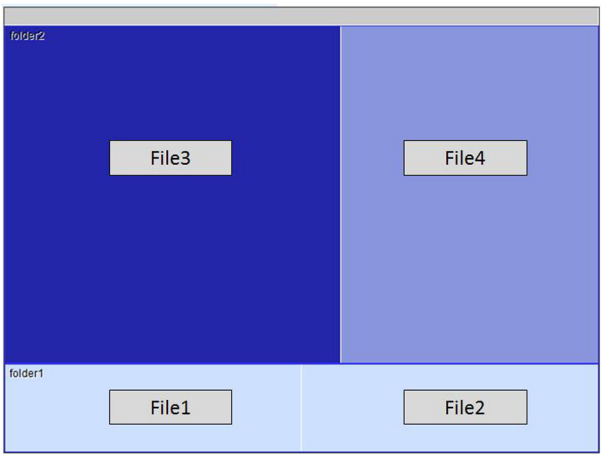


图 3 Treemap 代码重复检测结果可视化示例

对于层次很深并且文件个数很多的情况,如果显示所有的文件结点,用户难以看清每个结点的颜色,缺乏实用价值,而且这样的显示方式需要很多的时间渲染显示结果,导致软件性能低下。在文件层次很多的情况下,可以考虑使用 Zoomable Treemap(可缩放树图)。Zoomable Treemap 是带有交互性的 Treemap,其特点是一开始并不显示所有层次的数据,仅当用户选择某个数据结点时,才以该结点为新的根结点,继续显示下层数据。Zoomable Treemap 的优点是可以在较高层次上隐藏细节,让用户可以发现直接查看数据难以发现的问题。

图 4 是使用 Zoomable Treemap 进行层次数据显示的示意图。可以看出,放大子结点,显示出了更多细节。在交互方面,可以考虑在用户点击相应的子结点区域后,即进行子结点数据的钻取显示,在子结点数据钻取显示之后,当用户点击标题栏,即返回到父结点层次的数据显示。

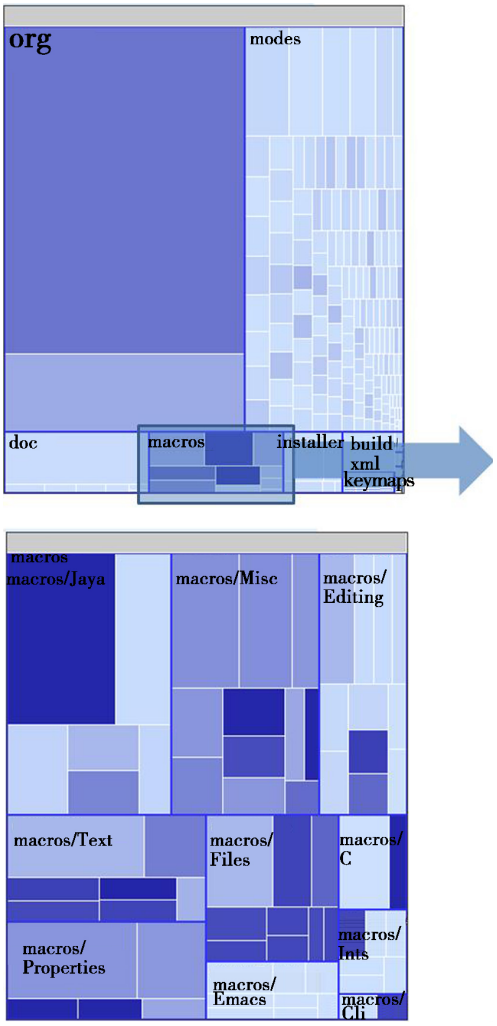


图 4 使用 Zoomable Treemap 进行层次数据显示

3 基于层次边聚合图的代码重复检测结果可视化

代码重复的检测结果集中,文件与文件夹之间有层次关系,同时文件之间还有代码重复关系。如果使用 Treemap 表示层次数据,而在 Treemap 的矩形块之间添加连接线表示叶子结点之间的关系,那么这些连线就会叠加在矩形块上,不但视觉效果

差,而且由于有的结点被密集的连线覆盖,不便于进行交互操作。Hierarchical Edge Bundles 是专门为表示层次数据的各叶子结点之间的网络关系而设计的[19-20],是一种较新颖的可视化形式,本文使用的中文名为层次边聚合图。这种可视化形式专用于表示数据结点之间有层次关系,同时数据的叶子结点之间有非层次的网络联系的数据集,可以解决连线覆盖结点的问题。

层次边聚合图使用圆环来表示数据[21],根据层次数据的总层次把圆环分成若干层,最内层表示叶子结点,而中间的圆形空间则为连线保留。在每层,用扇环表示各个数据结点[22],扇环的角度根据每个结点的值占总值的比重而定,内层结点之间用连线表示网络关系。图 5 是根据图 1 所示的典型代码重复情况生成层次边聚合图的示意图。

可以直观地看出,层次边聚合图减少了视觉上的复杂性,还可以通过颜色、材质等表示更多数据维度。层次边聚合图的不足之处是需要将环内的空间留作连线的显示,层次过多则环内空间变小; 因此它不适合显示层次过多的数据,而且由于层次边聚合图需要表现各结点之间的关系,从视觉上来说,它不是一个递归的结构,不便于像 Treemap 那样进行数据钻取。

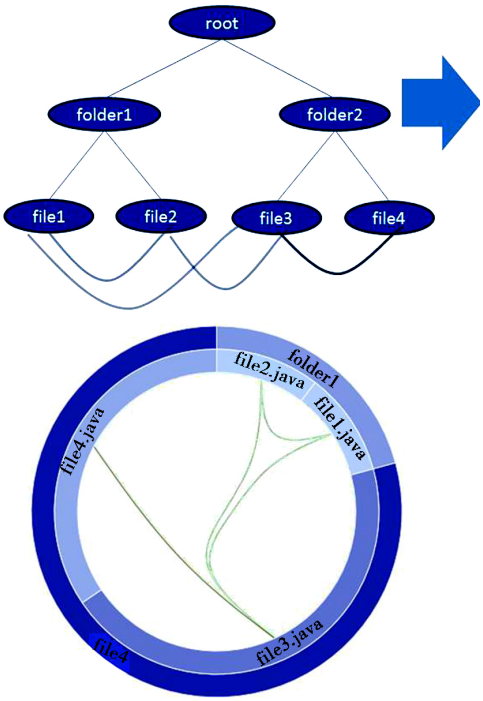
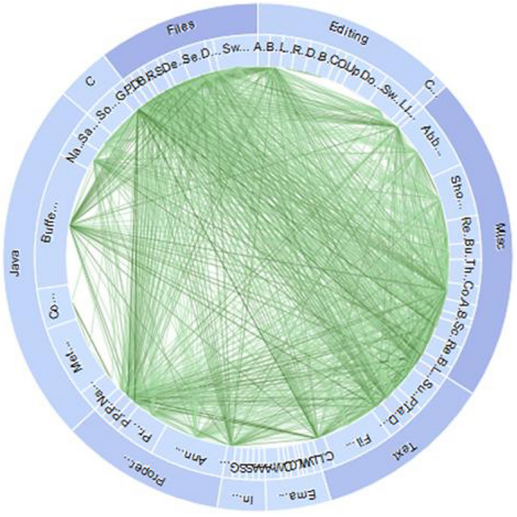
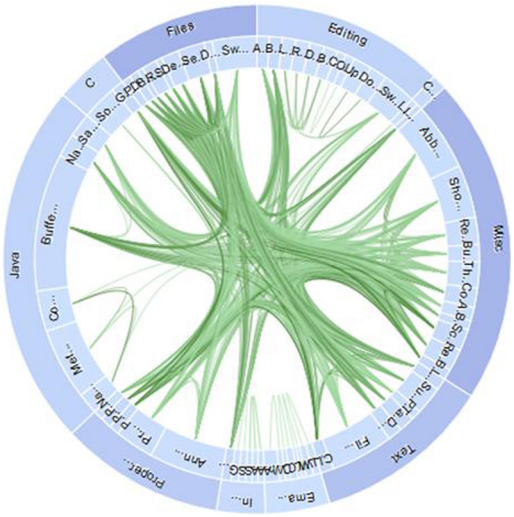


图 5　生成层次边聚合图的示意图

随着数据之间关系数目的增长,越来越多的连线几乎不可避免地会出现交叉或重复,而交叉或重复将增大图形的视觉复杂度,使结点之间的关系难以看清,如图 6(a) 所示。选择降低视觉复杂度的连线布局方式就变得非常重要了。层次边聚合图使用的方式是通过叶子结点的上层结点作为控制点生成样条曲线作为连接线[23],使相似的连接线聚在一起构成线束以降低连线的视觉复杂度[24],如图 6 (b) 所示。



(a) 直线



(b) 曲线

图 6　使用直线与曲线作为连接线的不同效果

层次边聚合图使用颜色的深浅表示重复度的高低。在 Treemap 中没有显示的结点,有可能因为与在 Treemap 中显示的结点有重复关系而需要在层次边聚合图中显示; 所以,当用户在 Treemap 中选择子结点数据之后,需要遍历选中结点下的所有叶子结点,找出与这些叶子结点有重复关系的其他叶子结点一并显示在层次边聚合图中。以图 1 所示的典型的代码重复为例,图 7 上边的 Treemap 是用户在原始的 Treemap 中选取 folder1 后的显示结果,只包含 2 个叶子结点 file1 与 file2,

下边层次边聚合图的显示的结果中包含 3 个叶子结点,其中增加的 file3 是与 file1 及 file2 有重复关系的结点。

如果只需要查看一个文件与其他文件的重复关系,可以在 Treemap 中仅选择一个文件,此时层次边聚合图将只显示包含这个文件的重复对,方便进行进一步分析。图 8 示出利用 Treemap 过滤数据,仅显示一个文件的重复关系的情况。

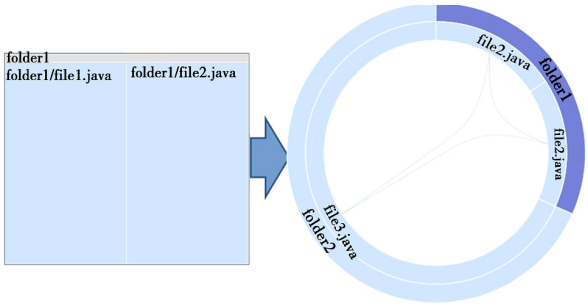


图 7 显示与 Treemap 中的结点有重复关系的结点

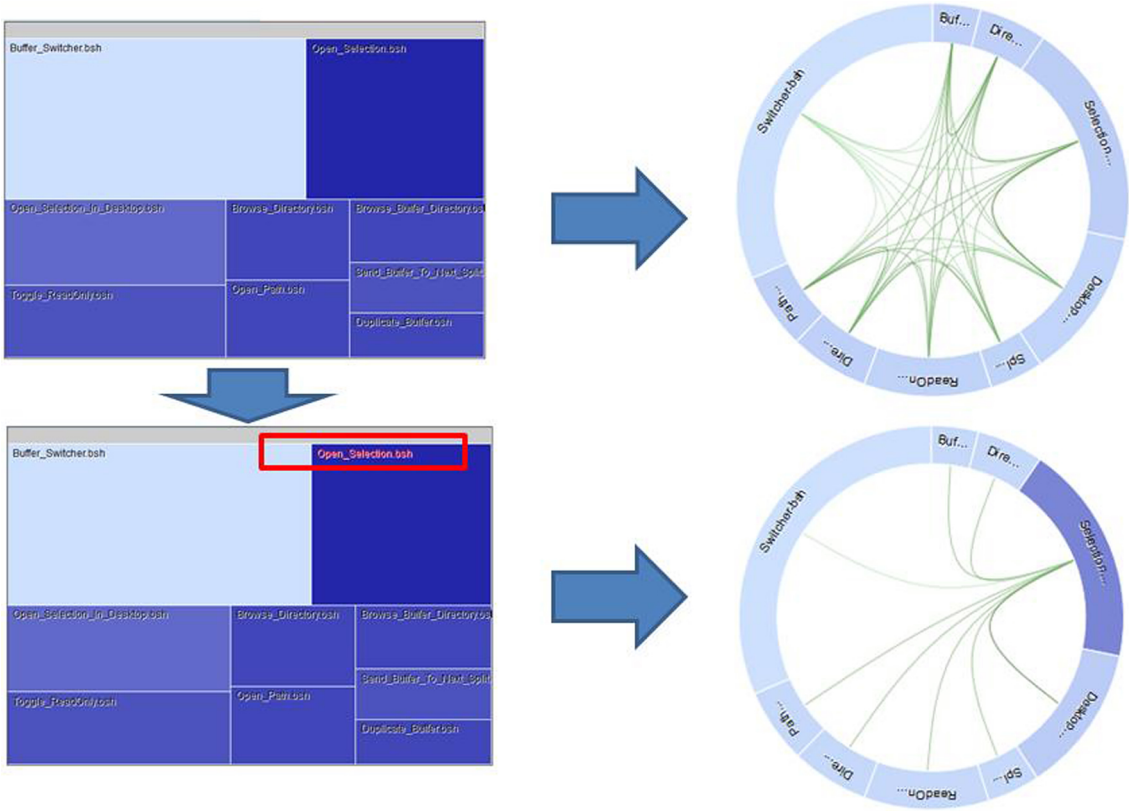


图 8 仅显示单个文件的重复关系

结点代表的文件名需要标注在表示数据结点的矩形块或扇环上,如果文字太长或结点空间太小,则不同结点的字符之间可能发生重叠,影响文本的可阅读性。改进的做法是当没有足够的空间可以显示全部文字时,仅显示文本字符串的前几个字符并加上省略号,当鼠标移动到标签上方时再显示全部文字。图 9 示出动态显示带省略号的标签上全部文字的例子。

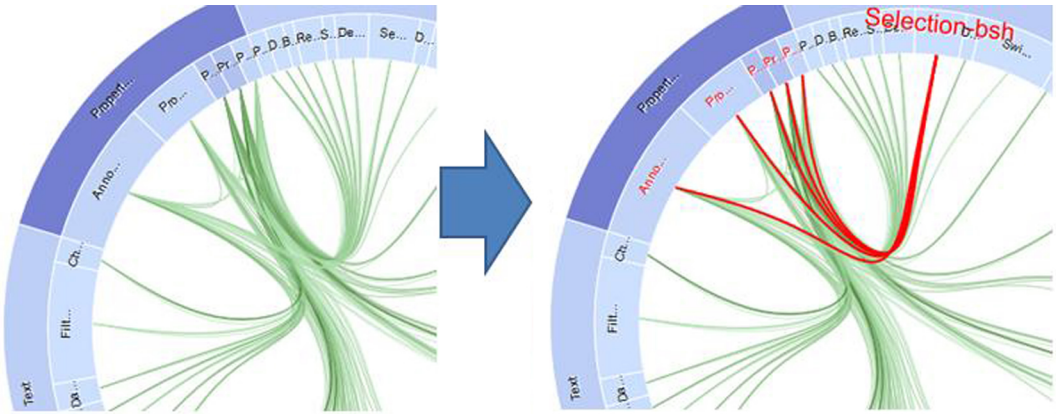


图 9 动态显示带省略号的标签文字

层次边聚合图用扇环显示结点，很容易因为结点的文字描述有旋转角度而不便阅读，为其加上自定义旋转功能可以方便读取结点上的文字。旋转功能可以通过鼠标的拖放来实现。图 10 示出在图 1 所示的典型的代码重复情况下旋转图形的效果。可以看出，有的文字旋转前在环的底部，不便于阅读，旋转后显示在环的上方，更易于识别。

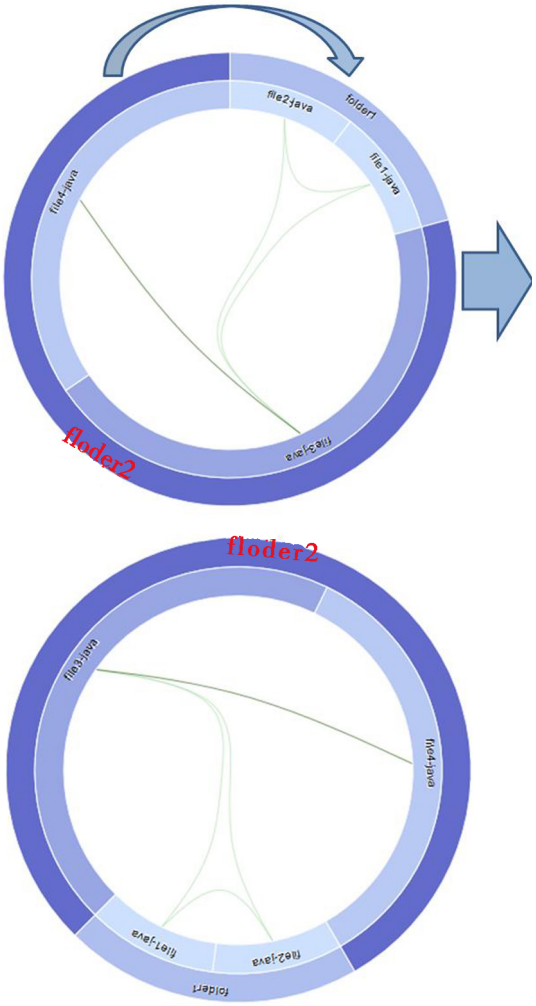


图 10　旋转可以改善文本的可读性

4　可视化原型系统 VizCD

信息可视化系统的设计，一般都要求做到直观化、关联化、艺术化、交互性，即功能性和易用性的统一。具体到代码重复的检测结果可视化，应该至少可以显示检测单元的详细信息及检测单元之间的关系，同时有良好的性能和可扩展性。

VizCD 的主要功能是进行代码文件的重复度检测，在获取检测结果之后，按照文件的重复对将数据进行转换，在主界面中以表格的形式显示所有重复对。在可视化过程中，用户可以在层次数据可视化形式和网络数据可视化形式之间选择，可以进行数据钻取操作显示过滤的数据。在可视化的界面中，用户可以选择自己感兴趣的重复对之后，返回主界面，以文件比较工具打开 2 个文件进行进一步操作。使用 Treemap 可以进行数据钻取操作，而层次边聚合图可以显示钻取后过滤的数据。

VizCD 的组件大致可以分为程序主界面、代码重复检测器、数据转换与过滤器、Treemap 可视化组件、Hierarchical Edge Bundles 可视化组件等几个部分。其中主界面具有以列表方式显示文件重复对的功能，可以启动可视化组件，还可以启动文件比较工具对 2 个文件进行比较和编辑。图 11 是 VizCD 的组件图。

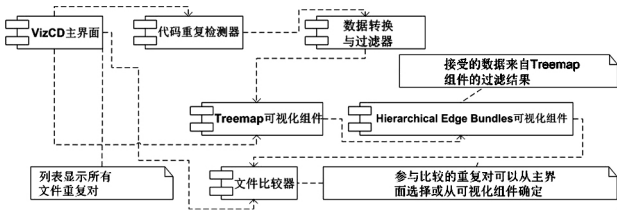


图 11　VizCD 的组件图

VizCD 的数据存储于文件中，包括源代码文件和代码重复检测得到的 JSON 数据文件。VizCD 的用户是数据流图中仅有的外部实体，外部实体针对数据的处理可分为代码重复检测和代码分析与编辑 2 个单元。代码分析与编辑单元还需要借助 Treemap 可视化和 Hierarchical Edge Bundles 可视化 2 个可交互单元进行重复对的过滤。图 12 是 VizCD 的数据流图。

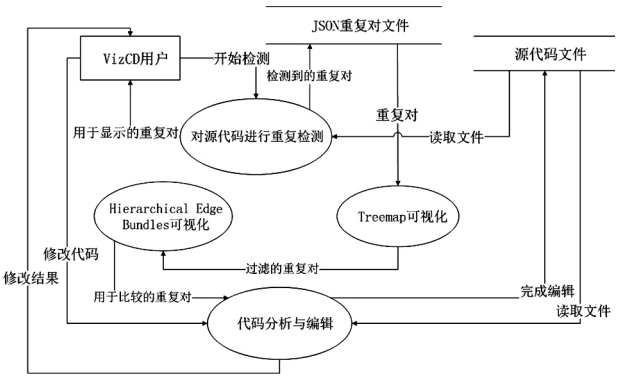


图 12　VizCD 的数据流图

VizCD 的典型使用流程可表示为图 13 所示的时序图。可以看出: 除主界面和检测结果之外，各个组件的生命周期都比较短，因为各组件大多是用作数据展示工具; 但是检测得到的 JSON 格式的数据可以使用文件保存起来。

VizCD 的可视化部分是使用 JavaScript 结合 SVG 实现的，由于 JavaScript 缺乏读写文件的能力，考虑到跨平台和数据处理的方便，VizCD 的主界面部分使用

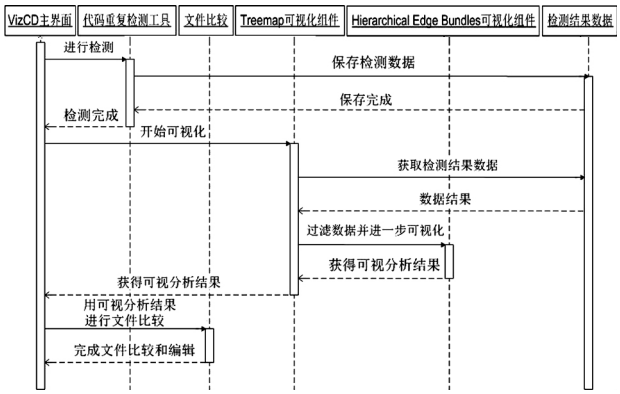


图 13 VizCD 的时序图

Java 语言实现,用户可在主界面上使用菜单打开可视化部分。图 14 示出 VizCD 使用不同的可视化组件时的运行截图。

主界面的 GUI 主要工作是实现功能选择菜单和对于文件重复对的列表,这部分功能可以用 Java 原生的 Swing 绘制。VizCD 中的可视化组件是 JavaScript 结合 SVG 实现的,选择的运行时组件是 Mozilla 的 XUL。JavaScript 是一种兼具面向对象和函数式编程风格的动态语言,已经成为 Web 应用开发事实上的标准[25]。

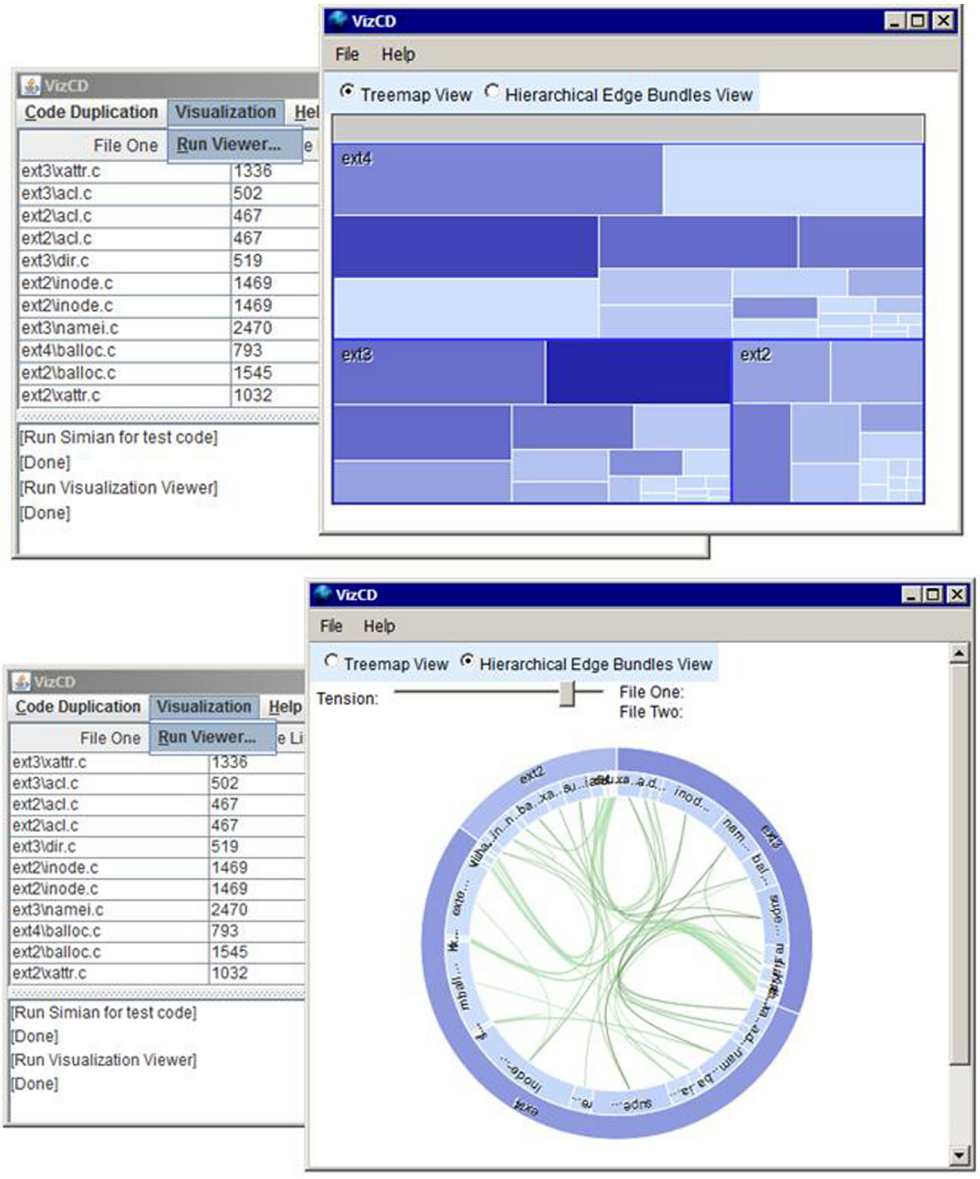


图 14 VizCD 使用不同的可视化组件的运行截图

jEdit 是一个使用 Java 语言开发的广受好评的开源代码编辑器。使用 VizCD 原型系统打开 jEdit 源程序所在的文件夹,可以看到 VizCD 已经自动进行了代码重复检测工作,以列表方式显示出共计 2 500多个重复对,包括每个重复对的文件行数和重复行数。

使用 Visualization 菜单上的 Run Viewer 功能打开可视化组件后,可以从 Treemap 上通过矩形的面

积和颜色的深浅直观地看到文件的大小和代码重复度的高低,通过点击可以放大 Treemap 上相应的结点查看细节。图 15 示出 Treemap 可视化结果。从 Treemap 中很容易得出的结论是: org 文件夹下的代码量最大,代码重复度也比较高; macros 文件夹下的代码量不大,但是代码重复度最高; modes 和 doc 文件夹下的代码量也比较大,代码重复度不高; 其他的文件夹下的代码量很少,代码重复度也很低。查看代码即可发现: org 是 jEdit 核心程序所在的文件夹,所以代码量很大; macros 文件夹下有大量用作宏的 BeanShell 脚本文件,相似功能的脚本文件之间往往只有很少的代码改动。

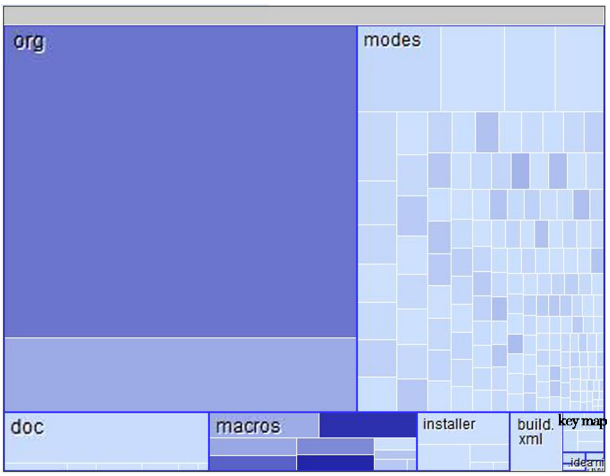


图 15　jEdit 代码重复 Treemap 可视化

切换到层次边聚合图之后,可以进一步看出代码重复关系的分布情况。从图 16 可以看到,modes 文件夹下的连接线非常密集。可见,这个文件夹中很多文件互相都有重复关系,这一点明显与其他几个文件夹的代码重复分布不同。查看代码可发现,

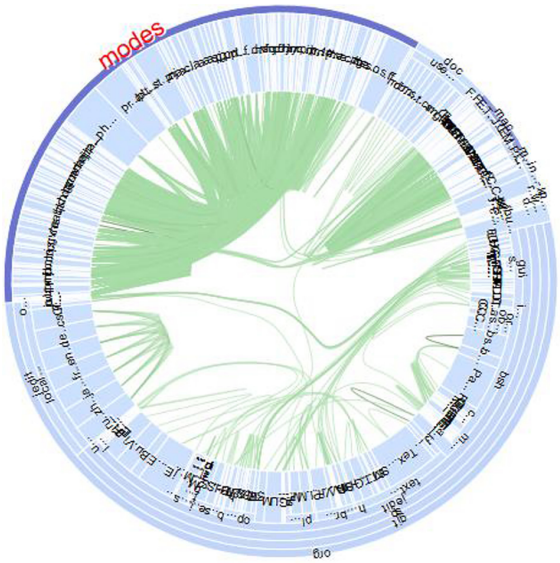


图 16　jEdit 代码重复层次边聚合图可视化

modes 文件夹下定义了很多用作不同程序设计语言关键字高亮的语法文件、相近的程序设计语言,例如 Java 和 JavaFX,语法的重复多,反映在语法文件上的代码重复也多。

另一个容易通过层次边聚合图观察到的现象是 org、macros、modes、doc 等文件夹下的代码重复基本都发生在文件夹内部的文件之间,比如 org 文件夹中的文件与 modes 文件夹中的文件就几乎没有代码重复。不同模块的代码实现的是不同的功能,所以不同模块之间的代码重复很少。这一现象反映了 jEdit 有良好的模块化设计,通过对 jEdit 进行的代码重复检测结果可视化分析,已经可以发现代码重复的诸多特点,从而为重构代码、提高程序质量提供了便利。

5　总结与展望

本文以文件为检测单元,以重复对的可视化为目标,设计了具有合理的可视化形式的代码重复检测结果可视化方案,最终使用多种软件技术实现了代码重复检测结果可视化的原型系统 VizCD。此系统可以使用具有交互性的可视化形式分析和展示代码重复检测的结果。

在代码重复的检测中,可能出现以包、类、函数等为单位的检测结果。对于这些检测单元的可视化设计仍然可以借鉴以文件为单位的设计方法; 但是由于检测单元的粒度不同,可能具有文件所不具备的软件度量结果,比如不同的函数可能有不同的扇入与扇出数目,不同的类可能具有不同的面向对象度量结果,重复的类可能与设计不当的继承体系有关,这些结果在可视化过程中可能需要以新的形式来展现。

在实际的软件开发过程中,需要使用 SVN、Git 等源代码版本控制工具,不同版本的代码具有大量的重复; 但是不同版本的改动多少并不相同,很多时候开发人员需要比较大量的版本以确定一处改动的位置,而且不同代码分支之间的合并加剧了这种比较的难度。如果代码重复的可视化工具能够与源代码版本控制工具配合使用,可以在一定程度上提高软件开发人员进行版本比较的工作效率。

在文件层次关系可视化的实现中,可以考虑用 2.5D Treemap 取代 Treemap 作为层次数据的展现方式。由于软件度量指标的多样化,传统的 Treemap 一般仅能在叶结点使用颜色和区域大小表示 2 个维度的数据,表示能力相对有限,难以添加更多度量

指标[26]。2.5D Treemap 作为对 Treemap 的改进,主要的扩展在于将其三维化,引入高度作为对另一个维度数据的表示。由于三维对象之间容易互相遮挡,2.5D Treemap 在实现上可以通过允许平移、旋转等方式提高可用性。

代码重复的度量结果往往需要配合其他软件度量指标进行分析,这些指标应该是可以由用户选择的,在软件度量中,对于某个特定的度量单位,可能需要显示大量的度量指标。在这种高维数据的场合下,可以使用 Parallel Coordinates 作为显示方式[27],当用户选取指定的几个维度之后,再改用 Treemap 等其他可视化方式显示。

可视化系统功能和方式很多,在应用中可以根据实际需要和数据特点灵活选择,希望本文的工作能够为代码重复检测结果可视化提供一些新的思路。

参 考 文 献

[1] SMITH R, HORWITZ S. Detecting and measuring similarity in code clones [C]//Proceedings of the Proceedings of the International Workshop on Software Clones (IWSC). [S. l.]: [s. n.],2009.

[2] KIM M, SAZAWAL V, NOTKIN D, et al. An empirical study of code clone genealogies [J]. Proceedings of the ACM SIGSOFT Software Engineering Notes, 2005,30(5):187.

[3] RIEGER M, DUCASSE S, LANZA M. Insights into system-wide code duplication [C]// Proceedings of the Reverse Engineering, 11th Working Conference on. [S. l.]: IEEE, 2005: 100-109.

[4] RIEGER M, DUCASSE S. Visual detection of duplicated code [C]//ECOOP 98 Workshop Ion on Object-Oriented Technology. London, UK: Springer-Verlag,1998:75-76.

[5] SAHA R K, ROY C K, SCHNEIDER K A. Visualizing the evolutionof code clones [C]// Proceedings of the Proceedings of the 5th International Workshop on Software Clones. [S. l.]: ACM, 2011:71-72.

[6] YOSHIMURA K, MIBE R. Visualizing code clone outbreak: An industrial case study [C]//Proceedings of the Software Clones (IWSC), 2012 6th International Workshop on. [S. l.]: IEEE, 2012:96-97.

[7] JIANG Z M. Visualizing and understanding code duplication in large software systems [D]. Ontario: University of Waterloo, 2006.

[8] DUCASSE S, RIEGER M, DEMEYER S. A language independent approach for detecting duplicated code [C]//Proceedings of the Software Maintenance, 1999 (ICSM 99) Proceedings IEEE International Conference on. [S. l.]: IEEE,1999:109-118.

[9] BAXTER I D, YAHIN A, MOURA L, et al. Clone detection using abstract syntax trees [C]//Proceedings of the Software Maintenance, 1998 Proceedings, International Conference on. [S. l.]: IEEE, 1998: 368-377.

[10] BELLON S, KOSCHKE R, ANTONIOL G, et al. Comparison and evaluation of clone detection tools [J]. Software Engineering, IEEE Transactions on, 2007, 33 (9):577.

[11] HARRIS S. Simian-Similarity Analyser [Z/OL]. [2017-02-26]. http://www.harukizaemon.com/simian/.

[12] 张浩,郭灿. 数据可视化技术应用趋势与分类研究 [J]. 软件导刊, 2012, 11 (5):169.

[13] TAIRAS R, GRAY J, BAXTER I. Visualization of clone detection results [C]//Proceedings of the Proceedings of the 2006 OOPSLA Workshop on Eclipse Technology Exchange. [S. l.]: ACM, 2006:50-54.

[14] BEDERSON B B, SHNEIDERMAN B, WATTENBERG M. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies [J]. ACM Transactions on Graphics (TOG), 2002, 21 (4):833.

[15] CHINTALAPANI G, PLAISANT C, SHNEIDERMAN B. Extending the utility of treemaps with flexible hierarchy [C]//Proceedings of the Information Visualisation, 2004 IV 2004 Proceedings Eighth International Conference on. [S. l.]: IEEE,2004: 335-344.

[16] VLIEGEN R, VAN Wijk J J, VAN der Linden E-J. Visualizing business data with generalized treemaps [J]. Visualization and Computer Graphics, IEEE Transactions on, 2006, 12 (5):789.

[17] TU Y, SHEN H W. Visualizing changes of hierarchical data using treemaps [J]. Visualization and Computer Graphics, IEEE Transactions on, 2007, 13 (6):1286.

[18] CHEN Yi, HU Haiyun, LI Zhilong. Performance compare and optimization of rectangular treemap layout algorithms [J]. Journal of Computer-Aided Design & Computer Graphics, 2013, 25 (11):1623 (in Chinese).

[19] MAEDA K. Visualization of code clone detection results and the implementation with structured data [J]. World Academy of Science, Engineering and Technology, 2011, 51 (1):1200.

[20] 刘旭. 基于 D3 的层次边聚合图设计与实现 [J]. 西华大学学报(自然科学版),2017,36(4):27.

[21] CORNELISSEN B, ZAIDMAN A, HOLTEN D, et al. Execution trace analysis through massive sequence and circular bundle views [J]. Journal of Systems and Software, 2008, 81 (12):2252.

[22] CORNELISSEN B, HOLTEN D, ZAIDMAN A, et al. Understanding execution traces using massive sequence and circular bundle views [C]//Proceedings of the Program Comprehension, 2007 ICPC 07 15th IEEE International Conference on. [S. l.]: IEEE,2007: 49-58.

[23] HOLTEN D, VAN Wijk J J. Force-directed edge bundling for graph visualization [J]. Computer Graphics Forum, 2010, 28 (3): 983.

[24] HOLTEN D. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data [J]. Visualization and Computer Graphics, IEEE Transactions on, 2006, 12 (5):741.

[25] 刘旭. Chrome V8 引擎中的 JavaScript 数组实现分析与性能优化 [J]. 计算机与现代化, 2014 (10):66.

[26] LÜ H, FOGARTY J. Cascaded treemaps: examining the visibility and stability of structure in treemaps [C]//Proceedings of the Proceedings of Graphics Interface. Windsor, Ontario, Canada: Canadian Information Processing Society,2008: 259-266.

[27] YUAN X, GUO P, XIAO H, et al. Scattering points in parallel coordinates [J]. Visualization and Computer Graphics, IEEE Transactions on, 2009, 15 (6):1001.

(编校:饶莉)