VT-Revolution: 交互式编程视频教程制作与观看系统

鲍 凌峰1, 邢 振昌2, 夏 鑫1, 王 新宇1, 李 善平1

1(浙江大学 计算机科学与技术学院, 浙江 杭州 310027)

2(南洋理工大学， 新加坡)

通讯作者: 鲍凌峰, E-mail: lingfeng@zju.edu.cn

摘 要: 开发人员可以直接从视频教程中观看到其中完成编程任务的工作流程。虽然现有工具可以较容易地帮助制作视频，但是探索编程视频教程中的工作流和与视频中的程序进行交互是极其困难的。本文设计并开发一个交互式编程视频教学制作与观看系统VT-Revolution. 该系统利用操作系统层面的插装技术记录教学视频作者在录制编程视频时的工作流记录，其观看系统提供一些列基于时间线上工作流与视频同步的交互界面来提高学习教学视频的体验。这些交互界面可以使观看者探索视频中的工作流和与视频中的代码和文件进行交互。本文还通过一个有90个开发人员参与的实验证明VT-Revolution系统可以有效的帮助开发人员学习教学视频中的编程知识。系统原型、演示视频以及相关实验材料可以通过访问网站<http://baolingfeng.xyz:8080/VTRevolution/> 查看。

关键词: 编程教学视频、工作流

# 背景介绍

编程教学视频一般展示一个编程任务极其相关的工作流。 Ponzanelli等人在一个40个开发人员参与的调查中发现有超过2/3的人每周或者每月会观看教学视频[1]。观看另外一个开发人员编程，比如在观看代码如何变动和生成，比传统的文本学习资源更有效[2]。使用视频捕捉工作软件（比如Snagit[3]）可以很简单地录制编程教学视频。一般来说，制作一个5分钟的视频教程比写一篇文字教程（通常会有若干张截图）要容易很多[4]。然而由于视频只是由一系列连续的截屏图像组成，这在几个方面限制视频观看者与视频内容进行交互：

* *缺少对视频中包含的工作流高层次抽象的整体视图。*一个编程任务通常涉及到多个文件和很多在文件上的改动。但是视频观看很难了解到任务完成过程的整体视图，比如何时以及如何改动某个文件，或者当前有哪些内容被添加到某个文件中。
* *无法支持对教学视频内容进行有效地搜索和浏览。*视频观看者只能看到某个时间点上视频中展示的文件内容；并且视频内容无法被搜索导致很难定位到包含观看者感兴趣内容的视频片段，比如视频中合适在某个文件中添加一个特定的API调用。
* *访问额外的补充学习资源不便捷。*编程教学视频中通常有很多观看者不熟悉的API，而且视频制作者很难详细解释所有的API，因此视频观看者通常需要一些额外的资源来辅助他们学习教学视频。但是视频内容无法通过直接粘帖复制用来搜索，导致观看者必须记住视频中的信息来手动搜索相关资源。

上述缺陷可能导致视频观看者对视频内容理解有误[5]、无法跟上视频节奏以及遗漏某些知识点[6]。一些研究[1, 7]通过OCR技术将视频内容转化成文字从而使视频可以被总结、搜索或者连接到其他资源。但是由于OCR技术非常耗时，这些工具一般会遗漏到很多截屏图像间的细小差别从而导致丢失一些重要信息，并且OCR转化的文本可能有很多错误。还有一些多媒体视频教程制作工具（比如Adobe Authorware[8]、 Adobe Director[9]）允许制作者创造可以交互的视频教程以及用更多丰富的信息去标注视频内容。但是这些交互是有限的，即观看者只能按照创作者定义的与视频进行交互，并且需要创作者额外的工作来标注观看者感兴趣的信息。

本文提出的交互式编程视频教程制作和观看系统的设计目标是：一方面，视频观看者可以自由的探索视频教程中的工作流和与教程内容进行交互；另一方面，避免视频教程制作者进行人工地标注视频中的工作流和其他额外的信息。为达成这一目标，本文提出VT-Revolution系统，其框架图如图1所示。VT-Revolution的视频制作系统在用屏幕捕捉软件录制视频时，同时使用一个叫做ActivitySpace的工具[10, 11]同步记录视频制作者在创建一个编程教程时的工作流。 ActivitySpace利用操作系统的API和Accessibility API记录视频创作者在各个应用上的一系列时间序列的人机交互动作以及与之交互的UI的内容。VT-Revolution视频观看系统通过一系列工作流历史数据的UI来帮助观看者探索视频教程中的工作流和与视频内容进行交互。

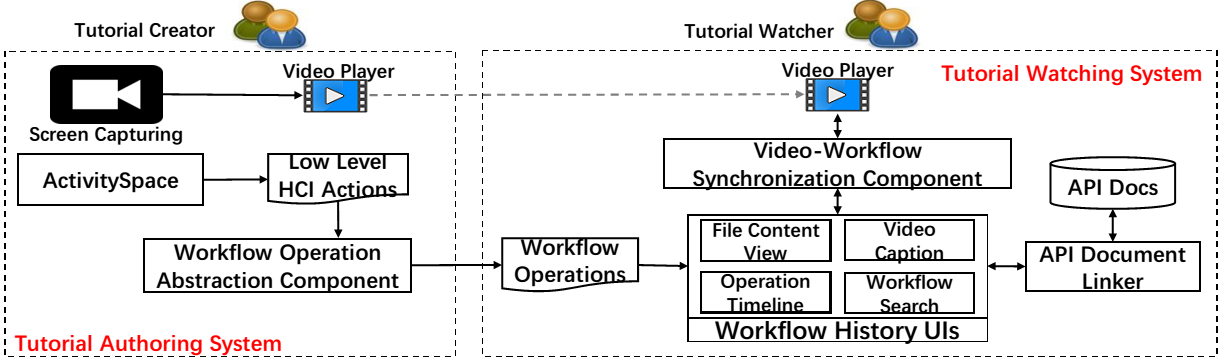


图1 VT-Revolution框架图

本文通过一个由90多个开发人员参与的实验来评估VT-Revolution系统。实验采用3个具有不同功能和复杂度的编程教学视频，参与人员需要观看这些编程教学视频来回答对应的问卷。通过分析回答问卷的准确率、花费时间以及参与者使用VT-Revolution或者直接使用视频播放器的满意度等，表明VT-Revolution可以更快更好地帮助开发人员学习视频教程。

# VT-Revolution系统介绍

## 视频制作系统

### 同步地视频录制与工作流记录

为了制作一个可交互编程视频教程，教程制作者只需要使用一个屏幕捕捉软件（如Snagit或者CamStudio）和ActivitySpace这个工具。 一般而言，屏幕捕捉软件都能通过快捷键来开始或者停止屏幕捕捉，比如Snagit使用Shift+F9和Shift+F10。 ActivitySpace能监控这类键盘事件，一旦探测到相关事件，它将开始同步地记录教程制作者在创建编程视频教程时的工作流。ActivitySpace不需要对任何应用程序进行特定的插装，它可以记录在很多常用软件上（比如IDE，文本编辑器，浏览器等）的用户交互行为。当用户与应用程序进行交互时，它能产生一个时间序列低层次的HCI动作流记录，其中每一个动作记录都包含一个时间戳、动作类型、基本的应用窗口信息以及当前焦点的UI组件信息。 UI组件信息可以通过Accessibility API获得，详细情况可以见本文作者对Accessibility API的一个调研[[1]](#footnote-1)。因为在编程教学视频中，制作者一般花费大部分的时间在IDE中演示如何完成一个编程任务，所以当前VT-Revolution原型系统主要考虑视频教程制作者在IDE中的交互动作。

### 工作流中的操作抽象

由于ActivitySpace记录的低层次的动作记录无法直接反映高层次的程序员编程动作，当前VT-Revolution系统抽象4类与编程任务相关的工作流动作：文件打开和切换、异常检查、代码元素添加和删除，以及文本内容编辑。

1. 文件打开和切换：VT-Revolution的观看系统从在ti-1时间之前的工作记录流中会维护一个不同文件的集合，如果时间ti的动作记录中的文件没有出现在这个集合中，则系统会产生一个文件打开操作*FileOpen<ti , name>*。如果在时间ti和ti-1的两个连续动作记录中文件不相同，则系统会产生一个文件切换操作*FileSwitch< ti , origin, target>*。
2. 异常检查：如果时间ti和ti-1的两个连续动作记录都是发生在“Console”（控制台）视图，并且视图内容包含关键字“exception”，这表明当前时间教程制作者在查看输出的异常信息，因此系统产生一个异常检查操作*InspectException< ti-1 , ti , exception>。*
3. 文件编辑：如果时间ti和ti-1的两个连续动作记录都是发生在IDE的文本编辑器上，并且动作包含的两个文件是同一个，那么系统将会尝试抽取出相邻时间内文件的改动。其中文件又分为两种类型：代码文件（文件后缀名为java）和非代码文件（其他后缀名，如xml、txt、properties）。

**文本内容编辑：**对于非代码文件，系统使用一个文本编辑工具来计算两个连续时间内相邻文件内容的变化，从而产生一个文本编辑操作*Edit< ti-1 , ti , file, change>，*其中*change*代表两个文件之间的改动。

**代码元素添加和删除：**对于代码文件，系统首先使用一个Java转化工具将代码内容转化为抽象语法树（AST）。如果两个相邻文件中的代码都是可以被转化的（可能由于代码不完整而无法转换为AST），系统使用一个简单的AST比较算法抽取出两个代码文件之间的代码变化，包含添加代码元素操作*Add< ti-1 , ti , type, info>*和删除代码元素操作*Delete< ti-1 , ti , type, info>*，具体的代码元素类型和信息见表1

表**1** 添加和删除代码元素操作

|  |  |  |
| --- | --- | --- |
| **ASTNode** | **Type** | **Info** |
| Import statement | Import | Package name  in the import statement |
| Field declaration | FieldDeclare | Field name,  Field datatype name |
| Variable | VarDeclare | Variable name,  Variable datatype name |
| Method call | MethodCall | Method identifier.  Object and its datatype  On which a method is call |

## 视频观看系统

图2 VT-Revolution观看系统截图. (1) 视频播放器 (2) 搜索工作流中的操作 (3) 展示工作流时间线 (4) 展示文件内容视图 (5) 教学视频中的编程任务介绍 (6) 视频字幕：当前时刻的操作 (7) 高亮的操作与视频播放是同步的 (8) 文件时间线：什么时候工作在哪个文件上以及时间长短 (9) 文件内容与视频播放是同步的 (10) 在工作流时间线上右键点击一个操作查看API文档

VT-Revolution观看系统包括多个界面可以帮助观看者很容易的探索工作流信息和与视频教程内容进行交互，界面截图见图2。下面介绍这几个界面的功能：

1. 视频字幕：这个功能可以提醒视频教程观看者“*当前时刻制作者在干什么*”。当视频播放时，系统会根据工作流操作同步地产生字幕。
2. 工作流操作时间线（timeline）：这个功能提供“*什么时候制作者在哪个文件上干了什么*”的整体视图。视图上的工作流操作按时间顺序从上到下排列。每个记录左侧的矩形代表操作的时间长短，同时还展示操作类型、涉及的文件。视频教程的时间线和工作流操作的时间线是同步的。当视频播放或者观看者浏览视频时间线时，当前的工作流操作会被高亮出来。在工作流操作时间线上双击一个操作会高亮这个操作并且会将视频的当前时间同步到这个操作的开始时间。
3. 文件内容视图：这个功能使观看者查看“*在视频中某个时间点前视频制作者操作文件的所有内容*”。视频中涉及的文件以一个标签视图展示出来，标签名即是文件名。当前的文件以及内容是与视频播放同步的，当在视频中文件内容改变时，视图中文件的内容也会相应变化，并且观看者还可以查看当前其他文件的内容。同时这个界面下方还有一个文件的时间线图用来表示“*什么时候制作者在哪个文件上工作以及工作了多少时间*”。
4. 搜索工作流操作：这个功能可以让观看者搜索包含他们感兴趣的代码元素的工作流操作。搜索会返回一个操作结果列表，点击其中一条记录可以将视频定位到所选择操作在视频所在的位置。
5. 查看API文档：在工作流操作时间线、文件内容视图和搜索工作流返回的结果列表上，观看者可以选中特定的代码元素然后查看相关的API文档。

# 实验

实验的目的是验证VT-Revolution系统是否能帮助新手程序员更有效的理解一个编程视频教程中的知识。

## 实验的过程

1. 首先我们用VT-Revolution系统制作3个编程视频教程，这3个视频教程涉及到不同的功能和编程复杂度，编程任务分别是Java email编程，mysql编程以及Eclipse插件开发，视频的时长分别为8分39秒，11分06秒以及19分19秒。
2. 接着我们从两个IT公司找了90个专业的程序员，这些程序员所在的项目组都不以Java为主要编程语言。这些参与者被随机分为6组，其中3个组为实验组，使用VT-Revolution系统观看一个视频教程，另外3组为对照组，只使用普通的视频播放器观看视频。实验被分为3个session，每一个session有一个实验组和一个对照组，他们都需要观看一个同样的视频教程。
3. 每一个参与人员在观看视频教程时，需要回答一个我们设计的问卷。问卷中的问题是根据视频教程中的知识点设计的。我们会统计每个参与者回答问卷的时间，答题的准确率。在完成问卷后，对于实验组的参与人员，我们还会让他们对VT-Revolution系统进行评价。

## 实验结果

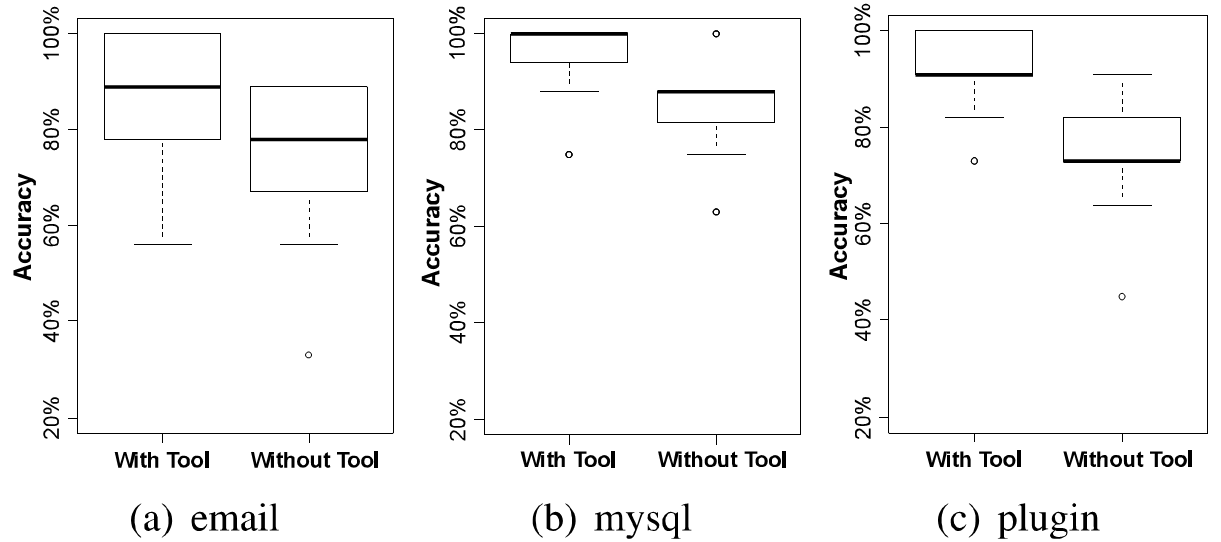


图3 回答问卷的准确率

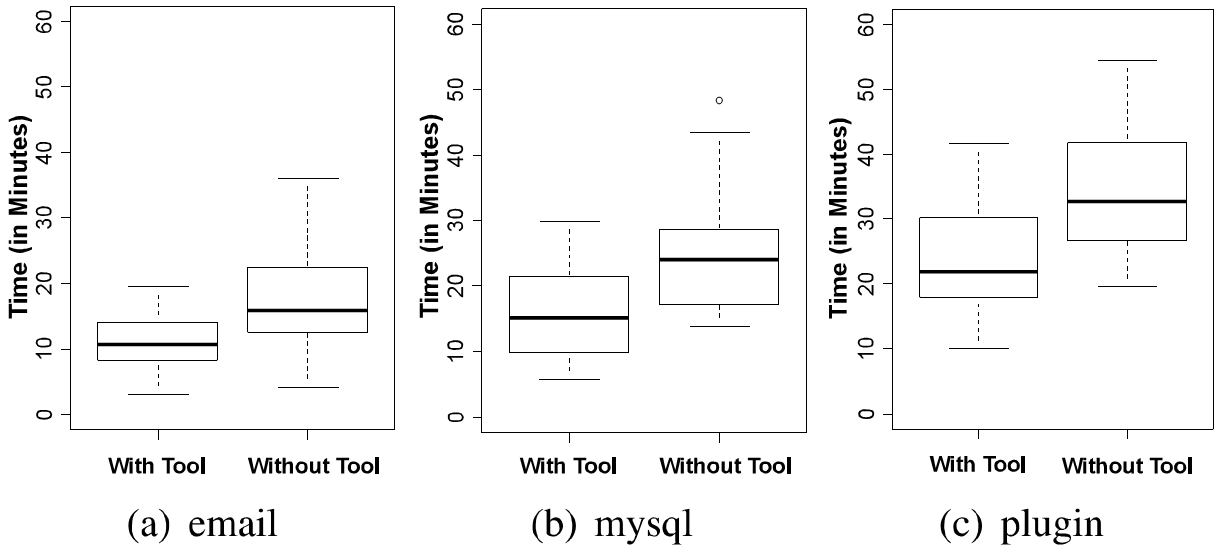


图4 完成问卷的时间

图3展示参与者回答问卷的准确率，实验结果表明实验组的平均准确率要比对照组高（email：0.89 vs 0.74， mysql：0.94 vs 0.84， plugin：0.91 vs 0.75）。 图4展示参与者完成问卷的时间，实验结果表明实验组完成问卷的平均时间比对照组短（email： 11.17 vs 17.16，mysql：16.05 vs 25.67，plugin：23.95 vs 34.81， 单位为分钟）。上述实验结果表明VT-Revolution工具可以有效的帮助参与人员学习编程视频教程中的知识。

# 总结

本文提出一个交互式编程视频教程制作与观看系统VT-Revolution来提高观看编程视频教程的学习体验。VT-Revolution系统将视频制作者的工作流与视频教程进行同步，可以让观看者获得高层次的工作流视图并允许观看者直接与其感兴趣的视频内容进行交互。通过实验验证VT-Revolution可以有效的帮助开发人员提高学校视频教程的效率。

References:

1. L. Ponzanelli, G. Bavota, A. Mocci, M. Di Penta, R. Oliveto, M. Hasan, B. Russo, S. Haiduc, and M. Lanza, “Too long; didn’t watch!: extracting relevant fragments from software development video tutorials,” in *Proceedings of the 38th International Conference on Software Engineering* *(ICSE)*. ACM, 2016, pp. 261–272L.
2. L. MacLeod, M.-A. Storey, and A. Bergen, “Code, camera, action: How software developers document and share program knowledge using  
   youtube,” in *Proceedings of 23rd IEEE International Conference on* *Program Comprehension (ICPC)*. IEEE Press, 2015, pp. 104–114.
3. “Snagit,” <https://www.techsmith.com/snagit.html>.
4. C. Plaisant and B. Shneiderman, “Show me! guidelines for producing recorded demonstrations,” in *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC’05)*. IEEE, 2005, pp. 171–178..
5. S. M. Harrison, “A comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphical user interface,” in *Proceedings of the SIGCHI conference on Human factor in computing systems (CHI)*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 82–89.
6. S. Palmiter, J. Elkerton, and P. Baggett, “Animated demonstrations vs written instructions for learning procedural tasks: a preliminary investigation,” *International Journal of Man-Machine Studies*, vol. 34, no. 5, pp. 687–701, 1991.
7. L. Bao, J. Li, Z. Xing, X. Wang, and B. Zhou, “Reverse engineering time-series interaction data from screen-captured videos,” in *Proceedings* *of the 22nd International Conference on Software Analysis, Evolution,* *and Reengineering (SANER)*. IEEE, 2015, pp. 399–408.
8. “Adobe authorware,” <http://www.adobe.com/products/authorware/>.
9. “Adobe director,” <http://www.adobe.com/products/director.html>.
10. L. Bao, D. Ye, Z. Xing, X. Xia, and X. Wang, “Activityspace: a remembrance framework to support interapplication information needs,” in *Proceedings of 30th IEEE/ACM International Conference on Automated* *Software Engineering (ASE)*. IEEE, 2015, pp. 864–869.
11. L. Bao, Z. Xing, X. Wang, and B. Zhou, “Tracking and analyzing cross-cutting activities in developers’ daily work (n),” in *Proceedings* *of 30th IEEE/ACM International Conference*

1. <https://baolingfeng.github.io/2016/01/01/accessibility-survey/> [↑](#footnote-ref-1)