专题：不断变化的 Web 应用程序

使用客户端重构实现个性化 Web 可访问性

通过 W3C 可访问性标准，大部分 Web 应用程序对于残疾人群都是不可访问且不可用的。开发者经常通过创建独立的可访问性应用程序来解决这个问题，但是那很少可用，并且通常提供的功能比原来少。另一个常用的解决方法是保持单一的应用程序，但是通过对每个请求的页面采用 FL 变换来创建一个无障碍的视图，是一个难以适用于所有用户的解决方案。这里描述的是第三种解决方案：让用户通过界面重构提高他们客户端浏览器的 Web 可访问性，它提供了许多单个应用程序定制的无障碍视图。

Alejandra Garrido,
Sergio Firmenich,
Gustavo Rossi,
andJuli án Grigera
阿根廷拉普拉塔国立大学

Nuria Medina-Medina
西班牙格拉纳达大学

Ivana Harari
阿根廷拉普拉塔国立大学

重构的初衷是提高软件内部质量的一种技术——如可理解性和可维护性，同时保留语义 1。在以前的工作中，我们采用了重构方法来提高 Web 应用程序的外部属性，如可用性 2。这些 Web 重构包括通过小的导航或界面转换来增强 Web 应用程序的感知性方面，比如用户互动和内容呈现方式，同时保留功能。重构也可以解决残障用户 3 的可访问性和可用性问题。还有，它对于解决所有观众的界面改进问题通常是不切实际的，因为残障用户在性质（视力障碍、认知障碍或运动障碍）、严重性（失明、色盲或斜视）和程度（完全或部分）上显著不同。在如此不同的上下文中，”所有人“是不可行的。

当应用重构来提高内部质量时，开发者决定应用什么改变，以及在哪儿应用，因为他们受益于改进。正如 Brian Foote 和 Joseph Yoder 所说，“当问题出现时，谁能更好地解决冲击每个设计问题的压力？谁又是不得不依靠这些设计的人？4”此外，不同的开发者或许更喜欢可替代的解决方案，对于相同的“坏味道”（即诱发重构 1 的设计问题）。根据这个普遍的哲理，我们相信那些终端用户能够根据他们自己的利益定制网站的界面。

我们建议授权用户（或者是类似代表）有选择的能力，在客户端浏览器，对自己可访问的每个站点进行页面重构。我们称我们的方法为重构（简称 CSWR）。CSWR 允许自动创建同一个应用程序的不同的、个性化界面来解决每个用户认为的特定的坏味道。（开发者应该继续致力于解决服务器端的一般的可用性问题，而且达到可访问性的最低

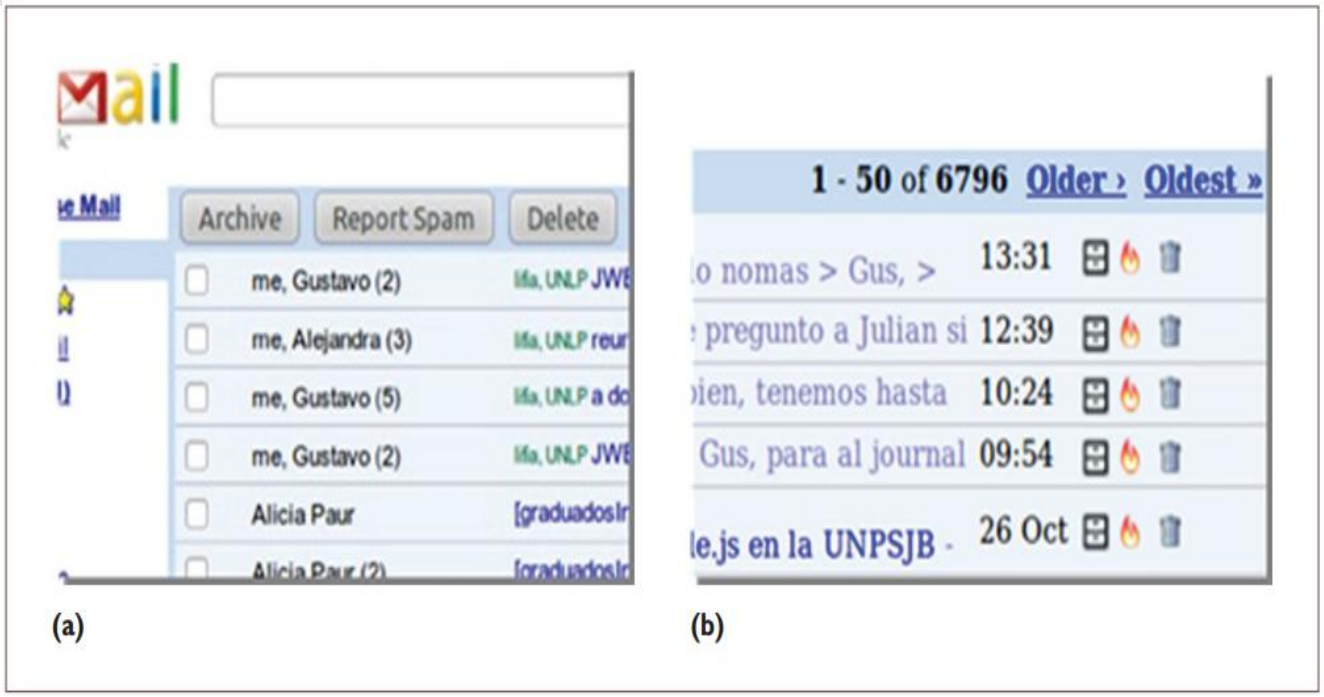

图 1：对 Gmail 应用重构。Gmail 界面前者(a)和后者(b) 分别应用分布式全局菜单来解决复选框和顶部操作上的可访问性问题。(Gmail 图标已得到转载许可)。

水平或“A”)。

在这里，我们描述 CSWR 和一个视障人士参与的实例研究（尽管类似解决方案可应用于其他残障人士)。

重构示例

图 1a 显示的是 Gmail（Google 的邮箱阅读器）的收信箱。Gmail 包括左侧的复选框，可以让用户选择多个邮件，这很方便对所有邮件进行操作。然而，对于使用屏幕阅读器的视障人士，不幸的是观众读完阅读栏之后不得不返回到阅读栏的顶部复选框去选择邮件，并且选择完邮件之后又不得不返回到顶部去应用此操作；他们报告这种现象为“坏可闻”（诱发重构的一个可访问性问题)。

诱发这种坏味道的一个重构是分布式全局菜单，它分配一个单独影响每个元素的元素列表菜单。这使一个操作的本地应用程序变得更轻松，因为当一个元素被阅读后需要仅仅一个简单的立即点击。图 1b 显示对 Gmai 收信箱应用这个重构的结果。这一系列动作是从顶部移走并且链接每个图标形式的邮件（每个对应一个可替代文本)。

然而，一些报告类似坏味道的用户能够更舒服地使用上下文菜单，所以这一系列动作不能被每个邮件读到。对于他们，上下文全局菜单重构更合适。而且有经验的用户更喜欢保持全局菜单，所以他们能一次操作多封邮件；对于他们，用推迟选择重构将移动复选框到最后一列。

客户端的 Web 重构

正如这个例子所述，Web 重构改变 Web 应用程序的导航结构或外观,去除可用性和可访问性的坏气味的同时保留它的内容和操作。在之前的工作中，我们用 Web 重构增强了开发生命周期 2,5 中的导航和演示。我们能通过系统应用和组合重构，根据特定的目标生成一个应用程序的全新版本--例如一个手机版本。在这里，我们提出一个相似的方法提高可访问性，即 Web 应用程序部署和实际应用期间使用重构，改变浏览器本身的界面。

我们 CSWR 方法有两个主要好处：

- *维护更简单* — 开发者维护一个单核心应用程序，其为解决普通观众问题应用 Web 可用性重构，当然不同的重构版本被创建且服务于不同的用户。
- *结构独立* — 开发一个 CSWR 需要很少（如果有的话）掌握目标应用程序的底层结构。

CSMRs 背后的引擎使用一个客户端的适应框架，旨在通过改变其 DOM 结构来适应现用的应用程序 6。CSMRs 通过实现类 AbstractRefactoring（由我们的框架提供）并且用重构机制重定义方法 adaptDocument()来实施。例如，考虑拆分页面重构，它通过将其划分成一组简单的页或段解决一个饱和、复杂页面的问题。在这种情况下，

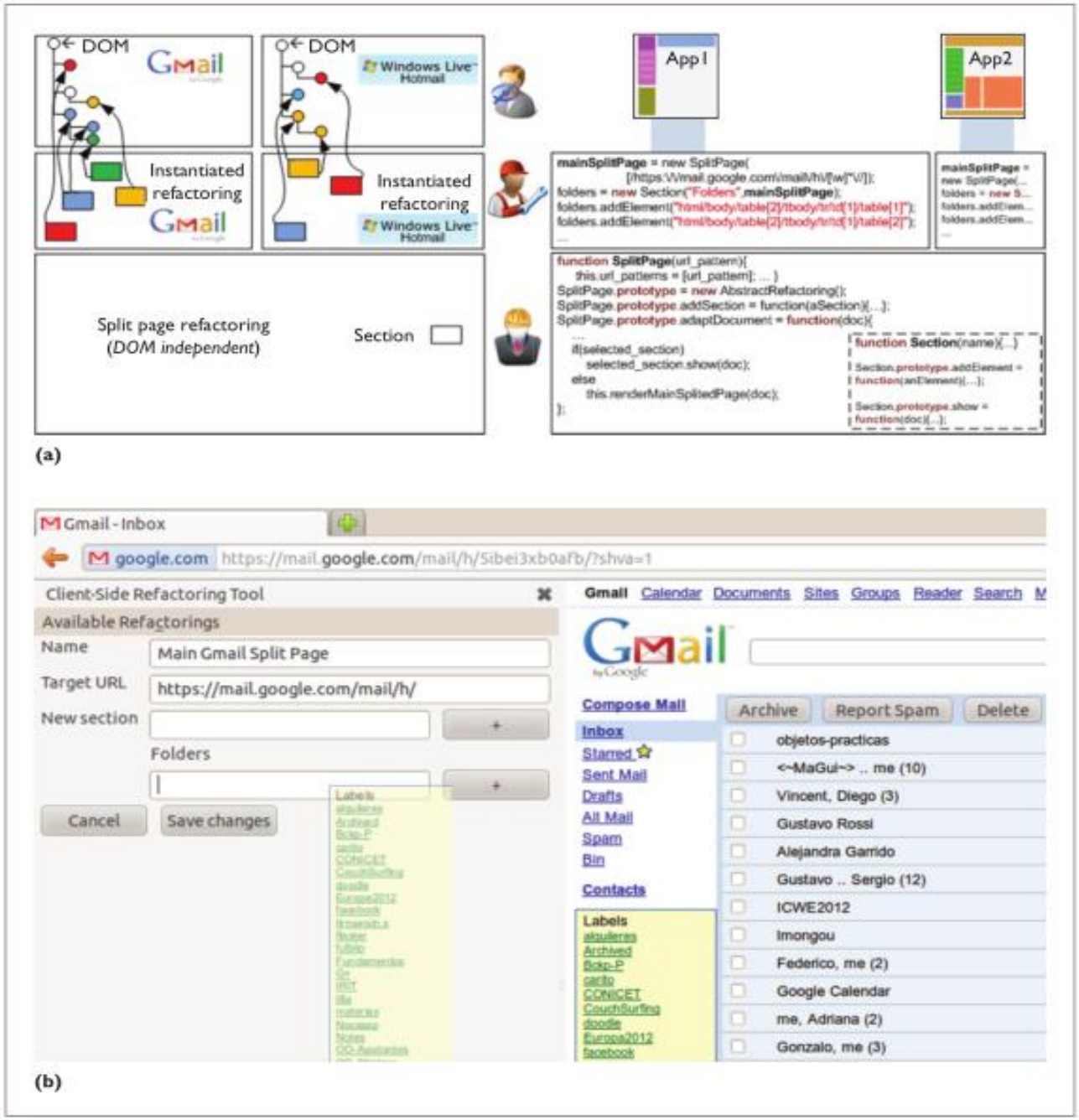

图 2：客户端网络重构。 a) 分页重构层次：底部是类的实现，中间是两个实例（对 Gmail 和 Hotmail），顶部是每个 DOM 结果。 b)分页实例：中介拖动 Gmail 标签来创建一个新的目录部分。

方法 saturated()接收代表不相关页面部分的 DOM 参数作为参数，并且创建为每个部分创建一个新
页面，使用新页面索引代替原页面的内容(参加图 2a 的底部)。

应用拆分页面重构到一个特定页面，你必须首先创建一个 SplitPage 实例(如图 2a 中部的两个实例)来传递参数

- 一个识别目标页的 URL 或一组相同站点页面的 URI 模式（如图 2a 代码中的所有 Gmail 页面）。和
- 类部分的实例，其包括通过 xPath 表达式指定的 DOM 元素。因为 DOM 元素可能并不总是通过基于属性识别的 xPath 查询指定的，来自 DOM 根的绝对 xPath 表达式或许被需要。

图 2a 中的三个等级对应重构过程中的三个步骤，其包括三个用户角色（图中中间列）：

- JavaScript(JS)程序员通过写一个参数化脚本创建新的重构，该脚本重用被我们框架的重构引擎提供的组件。
- 一个为特定网站的中介实例化重构。（JS 程序员也能扮演这个角色。）重构可通过写代码（如图 2a）或用我们的重构工具（如图 2b）实现实例化。这个图形化工具让目标页上的中介点击来选择作为每个重构参数值的组件。例如，图 2b 显示 SplitPage 的一个实例，这里邮件标签组被拖到进入新一页的目录部分。
- 最终用户通过从他们 Web 浏览器中的无障碍菜单选择来安装实例化重构；从此，他们可以访问按他们需要配置的重构网站。不同于传统的重构，我们添加新步骤（从实例中分离）使不能写脚本或用图形化重构工具的残疾用户也能通过选择他们自己的重构来成为过程的一部分。

CSWR 的动力不仅仅来自于让最终用户选择特定重构，也来自于让中介用不同的方式组合重构。CSWR 在实例化时候完成组合，并且需要特殊处理，因为重构能互相干扰。类似于代码重构，一个应用重构或许在下一个重构的前提下无效。在这种情况下，CSWR 的前提是被指定为参数（xPath 表达式）的 DOM 元素实体。所以，如果应用重构改变识别目标元素的 xPath,它或许会使后续重构无效。例如，如果拆分页面和分布式全局菜单都在原来的 DOM 元素上实例化，并且拆分页面先被应用，分布式全局菜单不会在原来的 xPath 定位式中找到目标元素。

我们的重构工具帮助中介用户正确地构成 CSWR 以至于他们能创建和分配一个应用程序完整的、无障碍的版本作为 CSWR 实例组合。当一个中介用户选择若干重构去组合的时候，工具创建并推荐一个可能的序列，首先实行结构重构（如拆分页面），然后重构以适应特定的 DOM 元素（如分布式全局菜单），最后实行 DOM 独立重构（如用他们的 Alt 文本替换图像）。CSWR 因此被按顺序实例化，用户能指定特定互不干扰的 CSWR 进行独立化。有了这些信息，该工具创建可选的 CSWR 菜单，这样当最终用户安装一套 CSWR 组合是，他们仍然可以单独地选择启动或顶用独立的 CSWR。

Gmail 实例研究:无障碍 Gmail

我们开展了对视障用户的 Gmail 的 HTML 版本为例来验证我们的要求，重构提供了一个更好的体验，当他们与最终用户定制设和自己的专长、屏幕阅读器的选择、个人喜好等等。我们扮演 JS 程序员和中介机构的角色。

初步研究

我们首先开展一项有七个潜在 Gmail 用户参与的研究来测试我们的初步假设：我们重构的 Gmail 版本比原来的更方便和实用。在七个用户中，六个是失明的，一个有严重的视力缺陷。测试用户有各种电脑和 Internet 技能水平；最常用的邮件客户端和 Web 浏览器，尽管只有一个以前使用过邮件工具。

我们给用户以下任务：

1. 阅读并回复一封邮件。
2. 撰写一封邮件并将它发给几个人。
3. 搜索并删除一封特定的发送信息。

用户必须首先在原来的 Gmail 上完成任务，所以我们能检查忽略的坏气味，然后在一个重构版本中，检测解决的坏气味并决定第一次重构尝试如何改进或破坏用户体验。

选定的重构是拆分页面（为分区的邮件列表，标签列表和主要的 Google 菜单),分布式全局菜单（为邮件操作），整理成列表（为无编号的操作组），去除多余的操作（删除顶部邮件操作的全局菜单及保留在邮件列表底部）和推迟选择（移动复选框列到每个邮件行尾）。

在研究期间，我们用观察和问卷调查收集各种反馈。这产生了一些发现：

- 在屏幕阅读器使用方面不同的技能水平会导致不同的混乱；例如，旨在简化结构的重构对于新手用户非常有用，但是对于有经验的用户却是繁重的。
- 用户建议新的重构，要包含使用上下文菜单来代替分布式菜单。

  使用网页邮箱客户端的新用户不能在原来的 Gmail 完成任务，但是能在重构版本中外部援助情况下完成（这证明了我们在新用户方面的假设）。

用户在内容的组织和功能方面有积极的反馈。用非常好、好、一般、差的评级量表，在组织/功能方面得到五个‘好’和一个‘一般’的反馈，但是在易用性方面得分仅有两个‘好’和四个‘一般’，这帮助我们收集其他坏味道，并且提高为中介和最终用户的工具。

实际实验

根据我们的初步研究，我们进行了一个新用户的实验。本研究中的 10 个盲人用户，其中 3 个在操作 Gmail 上有经验，其他 7 个在 Web 浏览器上有经验，但对于 Gmail 没有经验。这次，我们的假设是一个个性化的 Gmail 版本比我

Table 1. Results comparing completely refactored vs. personalized versions of Gmail.

User	Selected refactorings	Completely refactored (secs)	Personalized (secs)	Drop rate (%)
1	SplitPage, ContextualizeMenu, Postpone Selection	180	160	11.11
2	SplitPage, ContextualizeMenu, Postpone Selection	300	200	33.33
3	SplitPage, ContextualizeMenu, Postpone Selection	143	43	69.93
4	SplitPage, ContextualizeMenu, Postpone Selection	91	52	42.86
5	SplitPage, ContextualizeMenu, Postpone Selection	68	66	2.94
			Partial	32.03
6	DistributeMenu, Postpone Selection	90	65	27.78
7	DistributeMenu, Postpone Selection	180	97	46.11
			Partial	36.94
			Overall	**33.44**

们完全重构的版本好。我们一次一个用户地进行测试，覆盖以前研究中的同样基础测试组的简化任务：

1. 删除特定发送者的所有邮件。
2. 在垃圾箱中找到一个特定的已删除邮件，并把它放回收信箱。
3. 回复任务 2 中恢复的邮件。

在实际实验之前，我们让用户在原来的 Gmail 中回复一个特定的邮件（如在任务 3 中）。这有双重目的：它让我们了解他们使用工具（浏览器和屏幕阅读器）的技能，并且减少没有 Gmail 经验的用户可能产生的偏差。

对于实验的主要部分，我们设计了基于以前研究经验的四种重构的最优组合，并将其应用在 Gmail 上：

- 分页来减少每个页面的内容，从而缓解内容访问；
- 分布式菜单，来简化应用在列表（如邮件）中每个列表项的任务；
- 上下文适应菜单，将一个列表项的当前行为作为一个下文菜单；和
- 推迟选择，让用户阅读邮件主题并立即检查它们之后对一些选定的邮件申请动作。

在我们要求测试者完成任务之前，我们解释了每个重构。一旦用户在完全重构版本完成任务，我们让它们安装它们自己的重构组合，其应用在它们浏览器上的菜单选项。然后他们再次执行受影响的任务 - 那些与选择的重构相关的任务 - 再做进一步比较。

结果

我们收集的主要测试法是任务完成时间，比较完全重构版本的时间与那些在个性化版本重复花的时间（如表 1）。在这 10 个用户中，5 个比分布式菜单更喜欢上下文适应菜单，2 个丢弃拆分页面，而剩余的更喜欢它本身的重构网站。

整体完成时间平均减少 33.44%，选择上下文适应菜单科目的减少 32.03%，没有拆分页面重构的组减少 36.94%。

从这第二次研究，我们收集新回馈，其产生以下的发现：

- 新手用户发现使用拆分页面导航更容易，但是这不适用于有经验的用户，因为拆分结构下一些任务需要额外的导航步骤（例如当目录被移动到一个单独的页面目录索引时）。
- 一些新手用户建议用一种新的方式拆分页面，这样一些功能总是存在的；其他人想根据所需更容易地隐藏主菜单。

提高 Web 可访问性的相关工作

理想情况下，可访问性应早期考虑，在 Web 应用程序设计中，并且网页应该遵循已有的标准和规范，诸如 Web 内容可访问性指南（WCAG）。例如 1,2 这样的指南能包括并应用在 Web 工程生命周期中。其他保证或强制执行可访问性的关键途径包括系统性评估指南规范 3 和自动检测网页的可访问性问题 4,5。尽管存在这些研究方法，但大多数 Web 应用程序还没有完全可访问，而这个问题必须用更加动态的方法来处理。

一个改造现有网页可访问性的著名技术是转码，其适用于动态转换，基于由开发者手动添加或自动从 Web 设计模型中获取的语义注释。转码可以应用于服务器、客户端或代理 6。我们的方法分享转码背后的哲学，但是大多数现有的用于可访问性的转码系统缺乏可扩展性和个性化。

- 所有的转码方法（包括文字放大和内容排序 6）都是由开发者预先定义的，而且它不可能添加新的转码方法。大多数这样的系统仅仅是可扩展的 – 根据什么样的网页将被转码 – 如果志愿者被允许注释网站和为未来参观者引用转码的话。我们的客户端 Web 重构（CSWR）方法允许一个新类型的志愿者（JavaScript 程序员），其可以添加新的重构来应对新的坏味道或者用不同的方法处理同样的坏味道。
- 转码意识的注释对所有用户有相同的影响，不管他们有什么特殊技能。因为转码从用户角度是显而易见的，它不可能根据每个用户对一个特定的网页适当微调。有了 CSWR，每个用户能为每个网站选择不同的重构组。
- 转码不一定保留行为；我们不能删除一些操作，例如当他们旨在简化内容的时候。相比之下，重构被视为行为保持的转换 7，其在 Web 应用程序的情况下，意味着保存内容和功能。
- 转码不一定会组合，甚至会相互干扰 6。我们提出 CSWR 组合作为另一种定制网站的方法，让用户增量地应用一系列重构。

兴趣在定制现有网页的客户端脚本上不断增加，其被使用 GreaseMonkey(www.greasespot.net)的大型机构证实，其是允许任何网页 DOM 变化的客户端脚本的一种流行工具。特定的工具诸如 WebAdapt2Me (http://www-03.ibm.com/able/accessibility_services/WebAdapt2Me.html)和专注于可访问性的 AccessMonkey10。然而，这两个工具仅仅让用户做出风格上的基本改变，如字体大小或颜色和内容顺序上的基本改变。

当涉及到可访问性的改进时，目前的客户端工具太落后。诸如 GreaseMonkey 的通用工具不提供脚本兼容机制；当不同的脚本被用在同一网页时，一个脚本的执行可以破坏接下来的脚本。而且，当 GreaseMonkey 让用户适应一个特定页面时，它不它们推广 – 也就是说，如果改变取决于 DOM 结构时，它们不能在不同的页面上应用同样的改变。尽管 GreaseMonkey 像织工一样出色，它也不能提供无障碍设施。相比之下，我们的工具是进一步提供重构定义、组成和安装机制的织工。基于客户端脚本、专为可访问性设计的工具，如 AccessMonkey，也有一些局限性，主要因为它专注于简单风格变化，这通常不足以解决诸如用户导航迷失或长导航链的问题。

参考文献

1. V. Luque Centeno et al., “Web Composition with WCAG in Mind,” Proc. Int’l Cross-Disciplinary Workshop on Web Accessibility (W4A), ACM, 2005, pp. 38–45.
2. P. Plessers et al., “Accessibility: A Web Engineering Approach,” Proc. 14th Int’l Conf. World Wide Web, ACM, 2005, pp. 353–362.
3. J. Vanderdonckt, A. Beirekdar, and M. Noirhomme-Fraiture, “Automated Evaluation of Web Usability and Accessibility by Guideline Review,” Proc. 4th Int’l Conf. Web Engineering, LNCS 3140, Springer, 2004, pp. 17–30.
4. C. Benavídez et al., “Semi-Automatic Evaluation of Web Accessibility with HERA 2.0,” Proc. Int’l Conf. Computers Helping People with Special Needs, LNCS 4061, Springer, 2006, pp. 199–206.
5. TAW3: Tool for the Analysis of Websites, Fundación CTIC, Spanish Ministry of Employment and Social Affairs (IMSERSO) Online Web Accessibility Test; www.tawdis.net.
6. C. Asakawa and H. Takagi, “Transcoding,” Web Accessibility: A Foundation for Research, S. Harper and Y. Yesilada, eds., Springer, 2008, pp. 231–261.
7. M. Fowler, Refactoring: Improving the Design of Existing Code, Addison Wesley, 1999.
8. A. Garrido, G. Rossi, and D. Distante, “Refactoring for Usability in Web Applications,” IEEE Software, vol. 3, no. 28, 2011, pp. 60–67.
9. O. Diaz, C. Arellano, and J. Iturrioz, “Layman Tuning of Websites: Facing Change Resilience,” Proc. 17th Int’l Conf. World Wide Web (WWW), 2008, ACM, pp. 127–128.
10. J. Bigham and R. Ladner, “Accessmonkey: A Collaborative Scripting Framework for Web Users and Developers,” Proc. Int’l Cross-Disciplinary Conf. Web Accessibility (W4A), ACM, 2007, pp. 25–34.

- 用户的习惯直接干扰结果。例如，有经验的用户直到他们尝试并使用它一段时间以后才能领会分布式菜单的益处，因为他们习惯处理全局菜单。

这些结果清楚地显示出个性化的重要性：因为有经验的用户能使用键盘组合快速地浏览页面，他们更喜欢加载页面和更简短的导航路径 — 一个阻碍有经验用户的解决方案，因为每当页面重加载都需要很多内容。

论述

在以前的工作中 6，我们开发工具让用户创建概念模型，然后基于 ModdingInterface 理念 7，定义在这些模型的适应性。我们现在正在规划为使用 CSWR 的用户专门适应这一概念。这个新的抽象层次会让开发人员在 DOM 元素上定义概念（及其性质），并且定义这一概念的适应性，而不是直接使用 xPath 表达式操纵 DOM 元素。所以，如果连个 Web 应用程序管理同一概念 — 从而形成共享相同抽象模型的一个应用程序家庭组 — 定义在这个抽象概念的 CSWR 能应用在这两个应用程序上。

例如，共享同一抽象模型的邮件应用程序（收件箱、文件夹、电子邮件等等）能用同一组定义在这个抽象概念的 CSWRs，但是它或许提高脚本弹性。这样的弹性是客户端脚本语言最重要的劣势之一，我们的方法也没有豁免：当网页的 DOM 发生变化时，脚本或许停止工作。如果开发者使用一个敏捷过程，服务器端重构会经常更新 DOM。 这种技术的另一个常见限制是他不适用于所有网站；例如，使用诸如 Flash 技术开发的网站可能产生问题。

尽管我们提出 CSWR 来提高视障用户的可访问性，开发者可以很容易采用同样的方法来创建 Web 应用程序的不同视图，其目的是提高其他外部品质或创建诸如手机版本之类的。注意 W3C 指南在可访问性（Web 内容可访问性指南；www.w3.org/TR/WCAG10）和移动（移动互联网最佳实践；www.w3.org/TR/mobile-bp）上有很多相似之处，如果他们起初没考虑在 Web 应用程序上，这也能实现为 CSWR。

重构是一个强大的、必不可少的工具，其让开发人员基于基于反馈提高运行的应用程序。这种反馈可能来自开发人员确定的代码中的坏味道，也可能来自于用户体验中发现的易用性和可访问性的坏味道。然而开发人员根据用户反馈纠正坏味道，特别是可访问性的坏味道，通常要花费很长时间，这通常不是一个优先的方法。所以我们让用户掌握重构，他们更清楚他们真正想要的东西。这不仅仅让用户定制特定的相互改善方案，而且消除了来自应用程序本身的主要开发周期的改善方案，从而降低了成本和精力。

Web 重构是动态改善用户体验的一个技术上不可抗拒的方法，因为它们是可重组的，并且让用户在不知道内部设计的情况下创建不同的应用程序版本。这有很大益处，因为它也允许一个使 CSWR 及其组合体可用的众包方法。当然，我们将来的工作会包括为志愿者创建一个众包工具，为一个特定网站加载新的公共重构或包含重构的实例，其包括一个组合重构包来创建一个网站的全新版本。我们建议拥有众包 CSWRs 以便为最终用户以尽可能小的负担来扩展使用。而且，为了克服因应对网页 DOM 演变造成存在不同重构版本的问题，我们的众包工具的结构能自动选择给定的 CSWR 或 CSWR 组的最新版本。

参考文献

1. M. Fowler, Refactoring: Improving the Design of Existing Code, Addison Wesley, 1999.
2. A. Garrido, G. Rossi, and D. Distante, “Refactoring for Usability in Web Applications,” IEEE Software, vol. 3, no. 28, 2011, pp. 60–67.
3. N. Medina-Medina et al., “Refactoring for Accessibility in Web Applications,” Proc. 11th Int’l Conf. Interacción Persona-Ordenador,Assoc.Interacción Persona-Ordenador, 2012, pp. 427–430;
4. B. Foote and J. Yoder, “Big Balls of Mud,” Pattern Languages of Program Design 4, N. Harrison, B. Foote, and H. Rohnert, eds., Addison Wesley, 2000, pp. 653-692.
5. N. Medina-Medina et al., “An Incremental Approach for Building Accessible and Usable Web Applications,” Proc. 11th Int’l Conf. Web Information System Eng.(WISE), Springer, 2010, pp. 564–577.
6. S. Firmenich et al., “A Crowdsourced Approach for Concern-Sensitive Integration of Information across the Web,” J. Web Engineering, vol. 10, no. 4, 2011, pp. 289–315.
7. O. Diaz, C. Arellano, and J. Iturrioz, “Layman Tuning of Websites: Facing Change Resilience,” Proc.17th Int’l Conf. World Wide Web (WWW), ACM, 2008,pp. 127–128.

Alejandra Garrido 是阿根廷拉普拉塔国立大学计算机学院的副教授，她是一名先进 IT 实验室(LIFIA)研究和开发的成员。她也是一名阿根廷国家科学技术研究委员会(CONICET)的研究员。她的研究兴趣包括重构和网络工程，致力于设计模式、框架、C 语言重构和可用性重构。Garrido

拥有伊利诺伊大学厄巴纳-香槟分校的计算机科学博士学位。她是 Hillside Group 的一名成员。可通过 garrido@lifia.info.unlp.edu.ar 联系她。

Sergio Firmenich 是阿根廷拉普拉塔国立大学计算机学院的教学助理，他也是一名先进 IT 实验室(LIFIA)研究和开发的成员。他的研究兴趣主要集中于 Web 应用程序的适应性，特别是工程方面现有应用程序的适应性。Firmenich 拥有拉普拉塔国立大学计算机科学的博士学位。可通过 firmenich@lifia.info.unlp.edu.ar 联系他。

Gustavo Rossi 是阿根廷拉普拉塔国立大学计算机学院的教授，和先进 IT 实验室(LIFIA)研究和开发的成员。他也是 CONICET 的研究员。他的研究兴趣包括 Web 应用程序设计和敏捷方法。Rossi 拥有巴西里约热内卢天主教大学情报学博士。他是面向对象超媒体设计方法(OOHDM)的开发者之一，也是 IEEE 和 ACM 的成员。可通过 gustavo@lifia.info.unlp.edu.ar 联系他。

Julián Grigera 是阿根廷拉普拉塔国立大学计算机学院的博士生。他的研究兴趣在 Web 开发和敏捷方法，并且他以前工作在上下文感知系统和检测机制，和 Web 应用程序和移动设备的可用性及可访问性。Grigera 拥有拉普拉塔大学的硕士学位，可通过 juliang@lifia.info.unlp.edu.ar 联系他。

Nuria Medina-Medina 是拉普拉塔大学计算机语言和系统专业的副教授和研究员，她是软件规范、开发、变革组织(GEDES)的一员.她的研究兴趣包括超媒体系统、用户建模、用户适应和软件变革，以及为视障人士重构的 Web 浏览器和生物信息学。Medina-Medina 拥有拉普拉塔大学的计算机科学博士学位。可通过 nmedina@ugr.es 联系她。

Ivana Harari 是阿根廷拉普拉塔大学计算机学院的副教授和 Web 可访问性方向主任。她的研究兴趣包括人机交互、移动用户界面设计和 Web 可访问性，以及可用性工程和测试，与用户为中心的设计，残疾人的自由开源软件（FOSS）工具和自适应、无障碍的移动接口。Harari 是拉普拉塔大学教学的教育专家。可通过 iharari@ada.info.unlp.ed u.ar 联系她。

被选择的 CS 文章和专栏也可在 http://ComputingNow.computer.org 免费获得。