计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(4):226−230 [doi: 10.15888/j.cnki.csa.007340]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

一种基于模式的界面生成方法①

崔　洛[1], 周千明[1], 魏　炜[2], 贺兴时[3]

[1](西安工程大学 计算机科学学院, 西安 710048)
[2](西安工程大学 应用技术学院, 西安 710048)
[3](西安工程大学 理学院, 西安 710048)
通讯作者: 贺兴时, E-mail: xsh1002@126.com

摘　要: 多设备环境下, 应用程序在不同设备上用户界面的差异性导致了界面设计工作的重复和困难. 应用界面模式, 开发者可以脱离使用繁琐的底层控件生成用户界面的开发方法, 专注于宏观的交互方案, 从而为多设备界面生成问题提供一个可能的解. 基于 PLML, 本文设计了一个设备无关的界面模式描述语言 SPLML 用于表示基于模式的界面元素信息, 实现了不同平台上的界面模式生成框架 UIPF 用于支持界面自动化生成, 并通过具体案例说明了该方案的可行性和有效性.

关键词: 界面模式; 多设备界面; PLML; 用户界面

引用格式: 崔洛,周千明,魏炜,贺兴时.一种基于模式的界面生成方法.计算机系统应用,2020,29(4):226–230. http://www.c-s-a.org.cn/1003-3254/7340.html

Pattern-Based Multi-Device User Interface Generation Method

CUI Luo[1], ZHOU Qian-Ming[1], WEI Wei[2], HE Xing-Shi[3]

[1](School of Computer Science, Xi'an Polytechnic University, Xi'an 710048, China)
[2](School of Applied Technology, Xi'an Polytechnic University, Xi'an 710048, China)
[3](School of Science, Xi'an Polytechnic University, Xi'an 710048, China)

Abstract: In a multi-device environment, the differences of applications on the user interfaces of different devices lead to duplication and difficulty in the interface designing work. With the application interface pattern, developers can get rid of the development method which generates user interface using tedious underlying UI controls, and focus on the macro-interactive solutions, thus providing a possible solution to multi-device interface generation problems. Based on PLML, a device-independent interface pattern description language SPLML is designed to represent the information of the pattern-based interface elements, and the user interface pattern generation framework UIPF on different platforms is used to support the automatic generation of the interface. The specific case illustrates the feasibility and effectiveness of the program.

Key words: interface pattern; multi-device interface; PLML; user interface

1　概述

界面生成在软件系统中占有重要地位. 当前, 由于计算设备的多样性和运行平台的差异性, 使得几乎每一款应用都需要在不同的设备上开发相应的客户端界面. 这导致了开发者会在不同平台上使用底层控件和布局工具重复构建同一应用的界面, 陷入到设计细节中.

为了解决跨设备、跨平台的界面可复用性问题, 一种基于模式的用户界面描述方案被广泛应用. 界面

① 基金项目: 陕西省重点研发计划 (2018XW-021); 陕西省教育厅科研计划 (19JK0377)
Foundation item: Key Research and Development Program of Shaanxi Province (2018XW-021); Scientific Research Program of Education Bureau, Shaanxi Province (19JK0377)
收稿时间: 2019-08-04; 修改时间: 2019-09-03; 采用时间: 2019-09-29; csa 在线出版时间: 2020-04-05

设计模式是针对特定界面设计问题, 通过总结前人设计经验, 抽象出的界面模型. 同时提供相关的自动化工具解析其界面模式, 从而生成符合各平台使用风格的具体用户界面[1]. 这样, 开发者可以从更宏观的角度把握整个用户界面的设计, 而不是拘泥在具体的控件选择和设计等细节中, 能够提高界面开发效率.

基于上述方案, 本文设计了一种界面模式描述语言 SPLML, 实现了一个基于模式的界面生成框架 UIPF, 可以运行在多设备上, 解析成符合各平台使用风格的用户界面, 从而提高界面设计和开发的效率.

2 相关工作

界面设计模式[2]是前人针对特定设计问题的经验的总结, 它隐含了界面的可用性设计, 可以对界面开发提供指导, 提高设计效率[3]. 当前的界面模式库仍然大多采用文本、草图、模板等非形式化方式而忽略界面模式的自动化应用. 这类研究包括《Designing interfaces》[4]、《The design of sites》[5]等书籍, 它们可以辅助设计者完成界面设计工作, 但应用中仍需要手工编写代码实现具体界面, 限制了应用的便利性.

另一类研究, 如 GUIDE[6]、Damask[7]等, 试图将模式和界面自动化设计工具结合起来, 除了研究模式集本身外, 还能够根据用户指定的设计需求自动选择界面模式. 有一些工具除了支持界面设计之外, 也能支持原型设计, 例如 MyUI[8]和 OlivaNova[9]等. 但这些工具都仅能支持某一个特定平台下的界面原型设计, 不能和具体开发过程很好的结合起来.

综上, 为了能在实际产品开发中高效的使用界面模式, 需要对其进行形式化的描述. 在当前的研究中, 主要使用 PLML[10]等基于 XML 的语言来进行形式化描述, 这是自动或半自动处理界面模式的基础. PLML 定义了一系列标签用于从不同的角度描述一个界面模式, 但其对支持自动化工具最重要的<implementation>标签并没有严格的形式化定义, 从而无法真正的支持使界面的自动生成.

针对此问题, 西北大学的华庆一、吴昊等做了有意义的探索[11], 对 PLML 的<implementation>标签重新做了设计, 并在此基础上设计了一个新的界面模式描述语言 GUIDPLM 用于支持将界面模式自动化. 但这些研究最终是通过在模式描述语言中嵌入具体的程序设计语言代码 (例如 Java), 以完成界面生成自动化. 这并不能减少界面重复设计开发的问题.

基于以上研究, 本文提出了一种基于 PLML 的模式描述语言 SPLML 描述界面模式, 并设计了一个界面模式生成框架 UIPF 支持不同平台上的实际界面设计与开发, 使得界面设计者不需要关注底层平台的界面控件, 从而形成一个较为完整的解决方案.

3 界面模式描述语言 SPLML

SPLML 借鉴了 GUIDPLM[11]的部分设计思想, 基于 PLML, 其改进点在于简化了 PLML 中部分与结构化描述无关的界面模式说明及描述性标签. SPLML 对<implementation>标签进行重新设计, 扩充了能够表示基础控件的新标签, 使其可以抽象的表示底层界面控件而又与平台无关. 最终由这些平台无关的抽象控件构成界面模式.

3.1 SPLML 的结构

SPLML 文档以 XML 为描述语言, 其根标签为<splml>, 内部可分为两部分, 首先是描述性部分, 包括四个标签<name>、<problem>、<context>、<solution>, 这部分采用半结构化描述, 其主要功能是让界面设计人员理解该模式要解决的问题和解决方案. 除了<name>标签外, 其他几个标签并不会影响到后续对 SPLML 文档做自动化解析. 第二部分是 SPLML 的核心, 即具体实现部分, 主要是标签<implementation>以及内部包含的各标签及属性. <implementation>标签中可以包含一些 SPLML 特有的抽象控件标签, 例如<menu>、<button>等, 分别表示菜单、按钮等控件, 这些控件又可以包含位置、颜色、甚至回调句柄等属性. SPLML 的结构如图 1 所示.

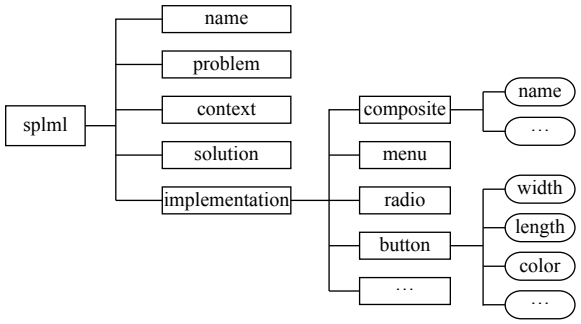

图 1 SPLML 的结构

这些可以添加到<implementation>内部的控件标签, 是我们对现有的界面设计模式及各平台常用的界

面基础控件进行总结的基础上定义的抽象控件标签, 在不同的平台上可以被解析成具体控件, 最终形成符合该平台风格的用户界面. 这些控件包含了菜单、按钮、卷滚条、文本框等常用控件, 可以满足绝大部分应用需求. 同时, 为了支持组合模式, 即在一个界面模式中直接使用其他模式, SPLML 支持<composite>标签, 此标签包含一个 name 属性, 其值必须为模式库中已有模式的名称, 否则该<composite>在界面自动生成时会被忽略. 普通控件和<composite>标签中表示的界面模式可以同时出现, 并按既定规则渲染到界面上.

SPLML 的部分 Schema 描述如下:

```
<xs:element name="splml">
<xs:element name="name" type="xs:string">
<xs:element name="implementation">
<xs:complexType>
<xs:sequence>
<xs:element name="textbox" type="xs:string" minOccurs="0"/>
<xs:attribute name="type" type="xs:string" use="required"/>
<xs:attribute name="chooser" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:element>
```

3.2 SPLML 举例

下面使用一个 SPLML 文档 (片段) 的实例阐述其使用方法. 本例中使用 SPLML 描述了 Dropdown Chooser[4](下拉选择框) 模式的使用, 这里只呈现<implementation >部分.

```
<splml>
<name>Dropdown Chooser</name>
<implementation>
<textbox type="input" chooser="calendar" action=" action"/>
</implementation>
</splml>
```

在<implementation>标签中, 只描述了 Dropdown Chooser 模式的核心组成部分, 即一个输入功能的文本框. 同时 chooser 属性值为 calendar, 表示点击文本框后出现的下拉选择框是一个日历, 根据需求, 这里还可以是时钟、表格等控件. action 表示待响应事件的处理方法, 本例中即选择了下拉框中具体日期后系统做出的处理, 可以由 UIPF 框架 (详见第 4 节) 解析并回调. 该模式的最终界面在不同的平台上有不同的显示风格, 但其基本功能是一致的, 体现了 SPLML 的跨平台性. 例如, 在 Web 环境下, 一个可能的展示如图 2 所示.

图 2 Web 环境下的 Dropdown Chooser 模式

4 界面模式生成框架 UIPF

为了能在实际开发中使用 SPLML, 我们设计了一个界面模式生成框架解析 SPLML, 并调用具体平台图形库底层控件最终自动生成用户界面. 由于 SPLML 的平台无关性, 我们在不同的平台上构建了相应的 UIPF, 设计人员将界面描述成 SPLML 后, 可以在不同的平台上被解析和使用, 进而生成符合各自平台风格的用户界面.

4.1 UIPF 的整体结构

UIPF 的整体架构共分为 3 层, 如图 3 所示.

图 3 UIPF 的架构

4.1.1 SPLML 解释器

顶层为 SPLML 解释器, 负责解析 SPLML 文档. SPLML 解释器本质上是一个 XML 解释器的扩展, 最终将 SPLML 文档中的界面内容表述成一个由 SPLML 各抽象控件所组成的有向无圈图, 称为 PG. SPLML 中描述的各界面模式 (或控件) 在 PG 中以叶子节点形式存在. 后续 UIPF 框架最终将 PG 渲染到界面上, 得到

具体的用户界面, 也可以根据用户行为刷新 PG, 完成和用户的交互. 这些工作由 4.1.2 节中的逻辑界面生成层完成.

4.1.2 逻辑界面生成层

中间层是逻辑界面生成部分, 为 UIPF 的控制核心, 分为 3 个子模块.

(1) 渲染控制器负责将 SPLM 解释器生成的有向无圈图表示成抽象控件的组合, 这些控件是 UIPF 定义的逻辑控件, 并不与任何具体平台相关. 同时它还负责在运行时维护界面模型与视图的一致性 (详见 4.2 节).

(2) 布局控制器的功能是控制整个界面的布局, 包括界面逻辑控件的大小、位置、外观等, 在 UIPF 中使用相对值表示 (具体实现中使用配置文件或样式文件管理). 这部分的功能是根据 SPLML 文档生成了不依赖于特定平台的逻辑界面. 在具体实现上, 布局控制器将整个应用的布局区分为整体布局和局部布局两部分, 且局部布局的优先级高于整体布局.

(3) 事件转换器的功能是将具体平台界面控件的底层事件转换成 UIPF 框架的内部事件, 使得系统中底层界面控件可以和上层 UIPF 框架建立统一的运行机制, 共同完成具体界面的渲染和构建. 详见 4.3 节.

4.1.3 底层界面控件

底层为具体的平台控件, 是各具体平台提供的基础控件. 在这一层中提供了对上层统一的接口, 可以很好的屏蔽 UIPF 框架对具体平台控件的依赖. 例如, 我们在 Web 环境下实现的 WUIPF 框架, 底层界面控件使用了 Vue.js[12]提供的功能. 在 WUIPF 中, 框架的上两层与底层交互时, 通过事件转换器将 UIPF 事件转换成 Vue.js 事件, 并使用了 Vue.js 的界面渲染机制构建出 Web 界面框架. 在其他平台 (如 Android) 上, 也可以方便的使用类似的方式构建 UIPF 框架.

4.2 UIPF 上模型与视图的一致性

UIPF 上使用被称为“动作”(action) 的一组实体保持模型 PG 和它的视图之间的动态一致性. 即用户操作应用的界面元素时, 会涉及界面呈现方式的实时变化, 而这种变化通过底层控件库的相应事件传导到 UIPF, 并被转换为 UIPF 的抽象事件 (详见 4.3 节), 同时促使 PG 做出符合用户期望的更新, 并及时刷新界面. 从用户观点看, 就是应用界面根据用户期望的方式实时更新.

UIPF 定义了一组动作用来维护视图与模型的一致性, 例如 RenderAction 表示重画 PG 上的节点, 进而刷新界面; CallbackAction 表示执行应用设立的回调函数等. PG 中叶子节点既可以表示 UIPF 定义的一个抽象控件, 也可以表示一个特定的界面模式. 而在 UIPF 中, 特定的模式是使用抽象控件组合而成, 对模式使用者而言, 其与原子的抽象控件并没有区别, 从而简化了框架的实现过程.

4.3 UIPF 的事件模型

UIPF 事件模型的设计独立于具体的应用平台. UIPF 定义了一组抽象事件及一个事件转换器, 用来与具体平台控件通信, 从而能够捕获并理解底层控件的各种事件, 并转换为 UIPF 能够理解的抽象事件. 这一机制可以驱动 SPLM 解释器生成的有向无圈图 PG 做出各种动作, 满足用户的交互需求, 并将结果反馈给事件转换器, 从而能够再次和底层控件通信, 最终完成用户界面的改变, 达到交互目的. UIPF 的事件模型如图 4 所示.

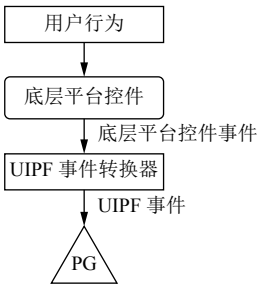

图 4 UIPF 的事件模型

例如, 我们针对 Web 平台开发的 WUIPF 框架原型中, 当用户做出交互行为时 (例如点击界面按钮), 用户的行为被浏览器捕捉, 转换为 Vue.js 事件, 经由 UIPF 事件转换器, 将其转换为 UIPF 能够理解的抽象事件, 而后驱动 PG 做出刷新, 后再将改动重新渲染到浏览器上, 从而完成这次交互.

5 应用案例

我们使用 SPLML 和 UIPF 实现了一个在线个人事务助理系统 ASSISTANT, 该系统基于 Web 和 Android 平台各自实现了客户端. 二者都使用了同样的 SPLML 文档描述界面元素, 并由基于不同平台的 UIPF 框架予以解析, 最终形成符合各自平台风格的用户界面.

如图 5 和图 6 所示, WEB 客户端由于屏幕尺寸大, 我们将各类任务列表尽量放在同一个页面上呈现, 减少用户切换页面的频率, 增强用户体验. 而在 Android

客户端中, 由于屏幕尺寸小, 我们将任务分屏显示, 滑动屏幕即可自由切换, 充分利用了移动客户端中的交互手势. 实验表明, 这组界面具有较高的扩展性和可维护性, 并能够方便的为用户使用.

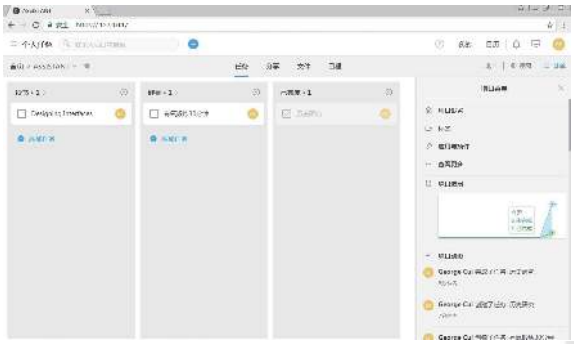

图 5 Web 客户端界面

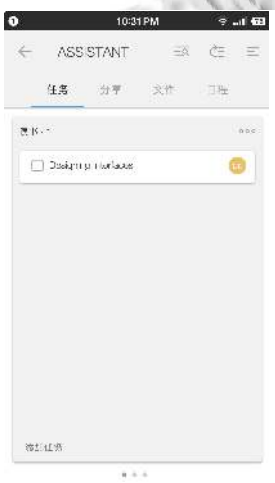

图 6 Android 客户端界面

在这个例子中, 我们使用了 Dashboard, Many Workspaces, Titled Sections, Liquid Layout, List Inlay, Button Groups[4]等大量界面模式, 它们在不同平台的 UIPF 解析和渲染下, 能够呈现出不同的, 但符合本平台交互风格的用户界面, 而这一过程中, 描述界面模式的 SPLML 是一致的. 这样就达成了只描述一次界面模式, 就可以到处运行的目的.

6 结论与展望

本文设计了一种设备无关的界面模式描述语言 SPLML, 并在不同平台上实现了 UIPF 框架用于支持各设备上基于模式的用户界面自动生成, 为减少建立多设备应用程序用户界面的设计工作提供了一种途径. 实验表明, 该方案是可行和有效的.

下一步的工作包括:

(1) 进一步扩充 SPLML 能够描述的模式库和虚拟控件, 使其能够处理更多的用户界面自动化场景.

(2) 支持 SPLML 中用户自定义标签的功能, 使得整套方案更具有扩展性和灵活性.

(3) 扩充 UIPF 框架, 使之能够支持主流的服务器端接口, 从而和 SPLML 相结合, 最终形成一个前后端统一的全栈解决方案.

参考文献

1 Dearden A, Finlay J. Pattern languages in HCI: A critical review. Human-Computer Interaction, 2006, 21(1): 49–102. [doi: 10.1207/s15327051hci2101_3]
2 Bayle E, Bellamy R, Casaday G, *et al*. Putting it all together: Towards a pattern language for interaction design: A CHI 97 workshop. ACM SIGCHI Bulletin, 1998, 30(1): 17–23. [doi: 10.1145/280571.280580]
3 姬翔, 华庆一, 李娟妮, 等. 基于界面设计模式的界面开发工具综述. 计算机应用研究, 2016, 33(10): 2889–2894. [doi: 10.3969/j.issn.1001-3695.2016.10.002]
4 Tidwell J. Designing Interfaces. 2nd ed. CA: O'Reilly Media, Inc., 2010.
5 Van Duyne DK, Landay JA, Hong JI. The Design of Sites: Patterns for Creating Winning Web Sites. 2nd ed. NJ: Prentice Hall, 2006.
6 Henninger S. An organizational learning method for applying usability guidelines and patterns. Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction. Toronto, ON, Canada. 2001. 141–155.
7 Lin J, Landay JA. Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Florence, Italy. 2008. 1313–1322.
8 Peissner M, Häbe D, Janssen D, *et al*. MyUI: Generating accessible user interfaces from multimodal design patterns. Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. Copenhagen, Denmark. 2012. 81–90.
9 Molina PJ. User interface generation with OlivaNova model execution system. Proceedings of the 9th International Conference on Intelligent User Interfaces. Funchal, Madeira, Portugal. 2004. 358–359.
10 Fincher S, Finlay J, Greene S, *et al*. Perspectives on HCI patterns: Concepts and tools. Proceedings of Extended Abstracts on Human Factors in Computing Systems. Ft. Lauderdale, FL, USA. 2003. 1044–1045.
11 吴昊, 华庆一. 改进的 PLML 在基于模型的用户界面开发中的应用研究. 软件, 2014, 35(7): 1–6. [doi: 10.3969/j.issn.1003-6970.2014.07.001]
12 Vue.js. https://vuejs.org/index.html. [2019-10-15].