计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(9):274−278 [doi: 10.15888/j.cnki.csa.005983]

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

国际化软件的硬编码和过度翻译问题自动测试研究①

刘雅君, 李爱民, 刘　晟, 袁　婷

(西安理工大学, 西安 710043)

摘　要: 伴随软件国际化开发技术不断发展的同时, 国际化测试的研究引起更多重视. 本文针对国际化软件中的硬编码和过度翻译两个问题, 研究并实现了自动化测试方案. 通过分析硬编码和过度翻译问题的概念、分类和测试方法现状, 本文总结现有人工测试的不足和 Struts2 国际化的技术特点, 最后结合被测项目的实际情况, 详细阐述了自动化测试方案. 该方案已经应用于某公司的 Web 产品的国际化测试中, 实践验证了对硬编码和过度翻译问题的自动测试的支持, 并取得了良好的效果.

关键词: 软件国际化测试; 硬编码; 过度翻译; 自动化测试

引用格式: 刘雅君,李爱民,刘晟,袁婷.国际化软件的硬编码和过度翻译问题自动测试研究.计算机系统应用,2017,26(9):274–278. http://www.c-s-a.org.cn/1003-3254/5983.html

Research on the Automatic Testing of Hardcoding and Over-Translation Problems in the International Software

LIU Ya-Jun, LI Ai-Min, LIU Sheng, YUAN Ting

(Xi'an University of Technology, Xi'an 710043, China)

Abstract: With the development of software internationalization technology, internationalization testing research is attracting more attention. In order to solve the problems of hardcoding and over-translation in international software, this paper proposes an automatic testing scheme. By analyzing the concept, classification and testing methods of the two problems, this paper summarizes the shortcomings of existing manual testing and the characteristics of Struts2 internationalization technology. Finally, combined with the actual situation of the tested project, the automatic test scheme is elaborated. The scheme has been applied to the internationalization testing of a real Web product, which is proved to be the support to hardcoding and over-translation automatic test, and has achieved good results.

Key words: software internationalization testing; hard coding; over-translation; automated testing

1　引言

早期的软件产品一般只能支持单一的语言, 且以英语为主要语言. 然而, 伴随软件技术的推广和普遍应用, 技术人员也开始积极考虑处理非英语语言的支持问题. 因此, 软件的国际化和本地化逐步引起重视.

在软件开发中, 国际化与本地化是指调整软件, 使之能适用于不同的语言及地区[1]. 其中, 国际化(Internationalization)是指将软件与特定语言及地区解耦的过程, 即软件设计和文档开发过程中, 使得功能和代码设计能处理多种语言和文化习俗, 能够在创建不同语言版本时, 不需要重新设计源程序代码的软件工程方法. 本地化(Localization)则是指当移植软件时, 使之与特定语言及地区耦合的过程. 国际化意味着产品有适用于任何地方的"潜力"; 本地化则是为了更适合于"特定"地方的使用. 对一项软件产品来说, 国际化只需做一次, 但本地化则要针对不同的区域各做一次. 这

① 收稿时间: 2017-01-04; 采用时间: 2017-02-13

两者之间是互补的, 并且两者合起来才能让一个系统适用于各种语言和地区[2].

针对国际化软件开发, 国际化测试的目的是测试软件的国际化支持能力, 发现国际化软件的潜在问题, 保证软件在世界不同区域中都能正常运行. 国际化测试工作一般包括四个级别[3]:

① 第一级国际化测试

测试英文版本的产品是否可以在非英语语言系统上正常运行.

② 第二级国际化测试

测试软件在任何区域设置中是否能正常运行, 并且是否支持本地化的字符输入, 输出和显示; 是否能按照本地化的习惯显示姓名、日期、时间、货币等数据等.

③ 第三级国际化测试

测试测试软件是否不需要重新设计或修改代码方便地进行本地化.

④ 第四级国际化测试

测试是否支持双向识别能力, 主要指对于阿拉伯语等双向语言的支持问题.

测试人员会根据项目资源和市场情况要求决定具体到某个级别的测试. 总之, 伴随全球化的趋势, 软件的国际化测试日益重要, 这将为软件本地化、软件维护、升级带来了极大的好处.

2 硬编码问题和过度翻译问题的研究

软件国际化的测试通常在本地化开始前进行, 以识别潜在的不支持软件国际化特性的问题. 理想的情况是, 国际化测试在英文版本完成时就已结束. 对于国际化软件, 硬编码和过度翻译问题是不可忽视的重要问题.

2.1 硬编码问题

硬编码问题是指将需要翻译的字符串没有提取到资源文件中. 早期在不考虑国际化编程的情况下, 显示在网页页面的的数据大部分都是硬编码在页面中的. 例如, 一个页面的标题可能采用如下的编码方式:

<title> Demo</title>

其中“Demo”就是以硬编码方式存在的. 正常的国际化下, 应该将这样的字符串提取到资源文件中. 根据本地浏览器区域设置, 通过国际化程序框架调用对应字符串显示. 类似于“Demo”这样的需要翻译的字符串有很多, 一般分为两类, 详述如下:

① 第一类硬编码问题

第一类硬编码问题是指不影响程序功能, 只影响用户界面显示的硬编码问题, 这类问题比较常见. 硬编码的字符串, 主要指的用户界面上显示的文字, 例如标签名, 按钮名称, 提示语等等. 这些文字本没有从程序中分离, 确实不利于以后软件本地化, 但是仅涉及前节所述的第三级国际化测试工作. 因此, 这类缺陷等级不高, 代码修改较易, 只需将这些遗漏的字符串将重新提取到资源文件, 请翻译专家翻译.

值得注意的是, 在这一类硬编码的问题, 不仅要测试页面文件, 而且要注意测试控制层代码. 例如在 JSP 项目中, Action 类中方法也可能存在这类硬编码问题.

② 第二类硬编码问题

第二类硬编码问题指的是会影响程序功能的硬编码问题. 这类问题一般级别较高, 涉及前节所述的第一级国际化测试工作. 这类问题数量较少, 比较隐蔽, 主要涉及系统变量被硬编码等情况. 例如在 Windows 系统中, 软件的默认安装路径“C:\Program Files”. 微软在做本地化操作系统时, 这些字符串在部分语言操作系统中进行了翻译. 因此, 如果这些字符串硬编码后, 将引起相关系统功能的失常.

2.2 过度翻译问题

事物有两面性, 如果字符串分离不当, 还会引起另一类问题, 称为过度翻译问题(Over translation), 主要涉及第一级和第三级国际化测试工作.

过度翻译问题是指将一些不需要本地化的内容提取到资源文件中, 从而可能引发用户界面问题或软件功能问题. 例如, 在页面首页提示语中包含时间数据. 如果不慎将这个时间和提示语中的其他字符一起提取到资源文件中, 则这个时间将不能根据系统的区域设置来进行变换.

2.3 测试方法现状及缺陷

针对上述问题, 主要仍然依靠人工测试完成. 例如第一类硬编码问题测试工作, 一般使用伪本地化加人工浏览的方法完成. 伪翻译(Pseudo Translation)是软件国际化测试的重要手段之一, 它不是软件真正本地化, 而是在源语言软件的基础上, 按照一定的规则, 将需要本地化的文本使用本地化文字进行替换, 模拟本地化软件的过程[4]. 这样可以在进行实际本地化处理之前, 预览和查看本地化的问题.

具体操作主要是对资源文件进行处理, 给资源文件中的每一个字符串加多个“#”字符组成前缀和后缀. 之后通过人工浏览界面, 确认运行软件是否出现的英文字符串. 这些英文字符串很有可能是硬编码缺陷. 在测试过程中, 要求测试人员对系统业务熟悉, 确保不遗漏界面, 对所有字符串认真检查.

第二类硬编码的测试依靠人工, 对测试人员要求较高. 因为这些缺陷不容易发现, 需要测试人员对系统业务熟悉程度高, 有丰富的国际化测试经验. 测试过程中, 测试人员根据软件功能确定需要重点关注系统变量, 并且更换特定环境进行测试.

过度翻译问题也是依靠人工测试. 有经验的国际化测试人员, 在了解项目业务和熟悉代码后, 总结相关测试点, 进行人工检查.

众所周知, 软件自动化测试是软件测试的一个非常重要的发展方向. 重复且工作量大的人工测试工作, 效率不高, 容易让人产生厌倦, 从而容易出错. 自动化测试则能很好的解决这类问题. 自动化测试能够弥补手工测试的不足, 提高测试精确度, 并且大幅度提高效率. 设计一种合理的自动化测试方案, 可以减轻国际化测试工组强度, 节省人力财力, 从而降低软件的测试成本.

3 硬编码和过度翻译自动测试方案的研究和实现

针对以上论述的硬编码和过度翻译测试的特征和其测试方法现状, 结合具体测试项目分析, 设计自动化测试方案. 本方案用 Java 语言实现, 通过静态测试实现对硬编码和过度翻译两个问题的测试.

3.1 测试项目的国际化方案

本测试项目是某公司的 Web 产品, 主要使用 Struts2 框架国际化技术. 该框架是一个基于 MVC 设计模式的 Web 应用框架, 是 Struts 的下一代产品. 它成功地结合了 WebWork 和 Struts1.x 两种 Web 框架. 在 MVC 设计模式中, Struts2 作为控制器(Controller)来建立模型与视图的数据交互.

Struts 2 国际化是建立在 Java 国际化的基础之上. Java 国际化的实现主要依靠“java.util.Locale”类和“java.util.ResourceBundle”抽象类. 其中, Locale 类用来提供本地信息, 主要包括语言、国家和地区等属性, 例如中国表示为. ResourceBundle 类称为资源包, 包含了特定于语言环境的资源对象. Struts2 框架对上述流程进行了进一步封装, 并且提供了灵活的资源包组织和加载方式.

3.2 硬编码和过度翻译的自动测试的方案

可疑的国际化测试点可能遍布软件的各个方面, 因此自动化测试的第一步需要对项目文件进行遍历. 根据 Struts2 国际化的技术特点, 结合被测项目的实际情况, 总结测试重点涉及页面文件(文件后缀为 Jsp)、校验器配置文件(文件命名规则为 Action 类名-validation.xml)、Action 类文件(文件后缀为 Java)和资源文件(文件后缀为 Properties).

本自动化测试方案使用 Java 语言实现, 具体流程如图 1 所示. 首先通过相应类库对上述文件进行解析, 提取文件中国际化敏感数据元素或方法. 针对这些元素或方法, 利用正则表达式进行检测, 从而确定可疑的硬编码和过度翻译问题候选项.

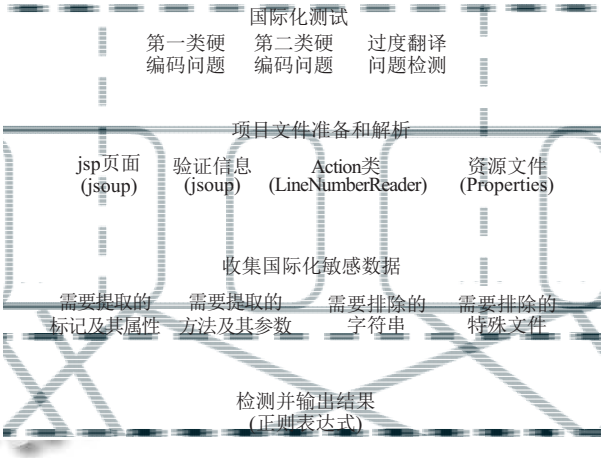

图 1 自动化测试方案总体框图

3.2.1 针对页面文件的测试

在页面文件中, 第一类硬编码问题比较常见. Struts2 框架的用户界面标签是在很常见的 Html 标签上额外添加前缀小号“s: ”, 主要分为两类:

① 表单标签

主要用于生成 Html 页面的表单元素, 以及普通表单元素的标签, 对应“s:label”, “s:submit”等.

② 非表单标签

主要用于生成页面上的非表单区域等, 对应“s:text”, “s:div”等标签.

正确的 Strust2 页面国际化, 主要是通过指定用户界面标签的 Name、Key 或者 Label 等属性进行的, 如:

<title><s:text name=“title” /></title>

<s:textfield name=“personName” key = “personName” />

<s:textfield name=“personName” label= “%{getText(‘personName)}” />

硬编码的情况是:

<title>示例</title>

<s:textfield name=“personName” label=“用户名” />

因此, 在自动化测试时, 查找页面文件并解析标签是首要任务. 我们选取了 Jsoup 工具进行解析. Jsoup 是著名的第三方类库, 其主要应用于解析 Html 文件, 获取用户需要的数据. 它提供了一套非常完善的 API 接口规范, 开发者可以通过 DOM 遍历或者类似于 JQuery 的操作方法来取出和操作数据. 同时它还包含了一个支持最新 Html5 技术的解析器分支, 能够很好的降低解析的时间和内存的占用.

使用 Jsoup 解析页面文件后, 将分以下三类情况进行检查:

① 测试页面文件的纯文本

将页面标签除去, 检查是否包含纯文本. 如果仍存在纯文本, 将与需要排除的字符串进行比对. 这些需要排除的字符串, 由测试人员预先设定, 如 URL、IP 地址、公司 Logo 等特殊字符串. 如果不能对比成功, 则认为该字符串是硬编码候选项并输出在测试报告中.

② 普通的 Html 标签的属性和内容检查

对普通的 Html 页面元素的内容和属性值进行利用正则表达式检验, 例如 Title 标记、Img 的 Alt 属性等.

③ Struts2 标签的属性和内容检查

如上述 Struts2 框架的用户界面标签, 不仅需要对 Struts2 标签的内容利用正则表达式进行检验, 而且要检查标签的 Name、Key 或者 Label 等属性.

当然, 在页面文件中, 也可能出现第二类硬编码错误. 这类错误的发现, 需要依赖通过预先设定国际化敏感的系统变量, 将硬编码候选项进行自动对比完成.

3.2.2 针对校验器配置文件的测试

页面上显示的校验错误信息也需要国际化, 主要涉及第一类硬编码问题. Struts 2 针对编码验证和验证框架都提供了国际化实现, 处理非常便捷. 在本测试项目中, 校验信息是通过校验器完成的. 例如检验密码的长度不小于 6 个字符, 正确国际化的实例为:

<field-validator type=“stringlength”>

<param name=“minLength”>6 </param>

<message key= “form.password.length”/>

</field-validator>

其中 Message 元素的 Key 属性, 就是验证错误信息在资源文件中的键值.

如果出现硬编码, 则会书写为:

<message>密码不能小于 6</message>

因此, 首要任务是搜索校验器配置文件和解析. 这些文件是以“validation.xml”字符串为后缀的 XML 文件, 他们的解析也是通过 Jsoup 工具完成的. Jsoup 提供了强大的 XML 操作能力, 高效又灵活. 通过 Select 方法和 Xpath, 程序将返回一个 Message 标签的 Elements 对象集合. 从而, 逐个检查 Message 标签中是否缺少 Key 属性, 即可得到硬编码候选项.

3.2.3 针对 Action 类的测试

Action 类文件也可能存在硬编码问题. 与页面文件相似, 这些缺陷多半是第一类硬编码问题, 极个别是第二类硬编码问题.

针对这些以 Java 为后缀的 Action 类文件, 主要采用普通文件读取方式, 进行逐行测试. 在 Action 类中, 想要正确地进行文本信息国际化, 一般需要使用“getText(String key)”方法. 这个函数将返回国际化资源文件中 Key 对应的值. 当然, 也有时通过 “getText (String key, String[] params) ”实现占位符填充.

因此, 自动化测试的重点是检验缺乏“getText”方法的字符串或输出语句. 这里的输出, 有涉及页面的直接输出, 也有涉及业务的异常错误输出. 另外, 第二类硬编码问题的发现, 与页面文件相同, 与测试人员的预先设定相关, 不再赘述.

3.2.4 针对资源文件的测试

Struts2 的国际化资源文件分为三种级别: ① 全局范围资源文件. ② 包范围资源文件. ③ Action 范围资源文件. 针对国际化资源文件的测试, 三类文件均需考虑其中的过度翻译问题. 下面是一个日期信息过度翻译的消息文本:

demo1 = 今天是 2017-1-1

正确的国际化方式, 日期不需翻译. 首先, 在资源文件中使用参数参数替换占位符{0}替代日期内容. 在页面文件中, 使用嵌套的 Param 标签来设置参数. 例如, 资源文件中消息文本为:

demo1 =今天是{0}.

页面文件代码片段为:

```
<s:text name="demo1">
  <s:param value="new java.util.Date()" />
</s:text>
```

Properties 属性文件内容都是以键值对形式存在的. 因此自动化测试时, 读取 Properties 文件后获取值的部分, 检查其中是否包含过度翻译的内容. 其中, 国际化敏感的数据作为测试的重点, 数据格式可能遍布资源文件的各个地方, 包括数字、货币、时间、日期、度量衡等.

4 结语

伴随软件国际化开发技术不断发展的同时, 国际化测试的研究也需要更多重视. 本文论述了国际化软件的硬编码和过度翻译问题的概念、分类和测试方法现状. 针对硬编码和过度翻译两个问题, 分析了现有人工测试的不足和 Struts2 国际化的技术特点, 结合被测项目的实际情况, 研究并实现了自动化测试方案.

本文提出的已经应用于某公司的 Web 产品的国际化测试中, 实践验证了对硬编码和过度翻译的自动测试的支持, 并取得了良好的效果. 当然, 国际化测试涉及的问题还有很多需要继续深入研究, 例如对数据库数据国际化的研究和国际化测试平台的研究等.

参考文献

1 董俊龙, 王武魁. 浅议基于 JSF 的 Java 国际化编程及其实现. 微计算机信息, 2009, 25(27): 170–171.

2 刘雅君, 徐进. 软件全球化测试技术的研究与实现. 计算机系统应用, 2010, 19(1): 40–45.

3 刘建国. 软件项目国际化和本地化的研究[硕士学位论文]. 北京: 华北电力大学(北京), 2013.

4 雷辉. 国际化软件测试研究[硕士学位论文]. 武汉: 湖北大学, 2007.

5 刘雅君. 软件回归测试技术. 计算机系统应用, 2011, 20(7): 156–159.