计算机科学2000Vol. 27№. 5

一种开发和测试国际化软件的新方法

A New Technique for Developing and Testing Internationalized Software

吴俊敏 安 虹 陈湘川 郑世荣
(中国科学技术大学计算机科学技术系 合肥 230027)

Abstract The internationalization of software is the current trend of software design. Compared with common process for software developing and testing, in order to develop internationalization software that can reliably run under multi-language environment with high quality, a large number of challenge problems need to be solved. This paper analyses new problems in the development of internationalized software and presents some new solutions. All these solutions have been adopted by Japanese Fuji-Xerox Inc. It not only saved the resource used for developing and testing internationalized software, but also speeded up the developing schedule and improved the quality of software.

Keywords Software internationalization, Localization, Software test

1 引言

软件的国际化是目前软件设计的一个趋势,一个软件要想在国际市场上取得成功,就必须使用当地的语言和一些约定俗成的数据格式。用户们希望一个软件能按照他们所习惯的方式来运行。例如,中国用户希望使用中文界面、日期表示为 yyyy 年 mm 月 dd 日,货币表示为圆;美国用户则希望使用英语,日期表示为 mm/dd/yy 的格式,货币表示为美圆($);而德国用户则希望看到用德语表示的信息,日期表示为 dd. mm. yyyy 的格式,货币表示为德国马克(DM)。不能"入乡随俗"的软件是很难让用户满意的。对软件公司来说,在开始投资开发出第一种语言版本后,只要能将转换到其他语言的费用控制得很低,那么在别的国家销售同样的软件所带来的利润是巨大的。

通常,软件测试总是在整个开发周期的最后进行,这种做法不利于产生高质量的软件。一旦在测试过程中发现错误,已经没有多少时间来修改它了。在国际化软件的开发中这种情况尤为明显,因为在开发期间要涉及开发、翻译和测试三方面人员,这些人可能并不在一处。本文给出了一些新的方法,使得测试贯穿于整个开发过程,从而可以在推出本国版本的同时推出其他多语言版本[1]。

2 国际化软件的开发

2.1 I18N 方法

目前流行的国际化软件设计方法称为 I18N[2]。其主要思想就是将与语言、文化、习俗相关的资源从主程序代码中分离出来;然后针对不同的语言和国家,再对这些资源进行本地化,就可以在多种语言环境下运行了。在各种语言环境下运行的都是同样的代码,只是资源文件不同而已。

我们用图1所示的例子来说明这种方法的思想。图 1a 是一个典型的未采用 I18N 方法编写的 C 语言程序。在该程序中,字符串"Hello World"和"Date"都被直接"固化"在源代码中。在采用 I18N 方法后,这些"固化"串都必须从主程序代码中分离出来。同时,与风俗习惯相关的其他数据格式,象本程序中的日期格式,也必须从主程序代码中分离出来。

图 1b 所示的是采用 I18N 方法后重新编写的代码。在程序代码中指定了一个 LOCALE 类,它定义了当前使用的语言、地区、字符集和排序方法等成员,每个成员随国家的不同而不同。对每种 LOCALE 还要提供相应的信息资源(Message Catalog),它包括程序中所有需要的字符串提示信息。一个国际化的程序允许用户随意指定一个 LOCALE,只要指定了一个 LOCALE 后,应用程序就应该以相应的语言和数据格式来运行。

2.2 开发过程

图2说明了采用 I18N 方法开发一个国际化软件的基本过程。整个开发过程大致分为以下四个阶段:

(1)编程人员按照 I18N 方法(利用 LOCALE 和 Message Catalog 来得到各种语言文化相关的信息)来编写代码;同时提供初始的与语言文化相关的信息资

源。

(2)将初始的语言文化相关的资源交给翻译人员，翻译成其他语言版本。

(3)编程人员把应用程序与初始的语言文化相关资源提交给测试人员，测试人员使用与初始的语言文化相关资源来测试应用程序。

(4)测试人员得到翻译过的语言文化相关资源，再次对应用程序进行测试。

```
Main(int argc, char*argv[])
{
    int month=6;
    int day=29;
    int year=1999;

    printf("Hello World\n");
    printf("Data:");
    printf("%d%d%d", month, day, year);
}
```

"国化"的字符串

"国化"的格式

a. 需要国际化的代码

```
Main(int argc, char*argv[])
{
    int month=6;
    int day=29;
    int year=1999;
    Locale*locale=GetLocale(locale string,      );
    Catalog*cat=OpenCatalog(locale,      );

    printf(GetCatalog(cat, MessageOne));
    printf(GetCatalog(cat, MessageTwo));

    printf(GetDateFormat(locale), month, day, year);
}
```

初始化 locale 和 Message catalogue

从消息资源中获取合适的信息

获取合适的日期格式

b. 国际化的代码

图1 采用 I18N 方法实现软件国际化的一个例子

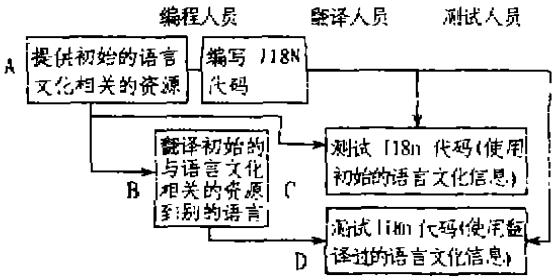

图2 开发国际化软件的一般过程

2.3 开发过程中需要解决的问题

生成国际化软件的思想在概念上是很简单的，尤其是当语言数很少，而应用程序又象例子程序那样简单时。但是，在实际开发一个能在十几种语言环境下正确运行的应用软件时，开发队伍就可能碰到很多问题了。在整个开发过程中，要涉及到三个不同的小组：编程人员，测试人员和翻译人员，各个小组都有一些问题需要解决。

从编程人员的角度来看，采用 I18N 方法后，只需要提供一个版本，他们的工作量大为减少了，但这中间出现的一个问题就是：编程人员难以确定代码是否已很好地国际化了，能否很好地支持各种语言文化资源，他们必须等待翻译人员提供翻译后的语言文化资源才能验证是否已正确实现了国际化的要求。

在翻译环节，翻译人员是语言专家，但他们有可能不熟悉该应用程序，有时甚至不熟悉整个计算机环境；同时，由于信息资源结构有时是很复杂的，这样在翻译时就有可能出现一些困难，导入一些新的错误。比如有些字符串(象 Window)在某种情况下是关键字，不能被翻译，而在另一种情况下又可能是提示信息的一部分，需要被翻译。在这种情况下就很容易将信息结构和关键字改动，从而发生错误。经常还有粗心带来的错误，比如将标识符 l 误变成标识符 ll。这种在翻译时产生的错误在测试时往往会出现一些很奇怪的现象，反馈到编程人员那儿时很难发现错误的根源。

对测试人员来说，国际化设计大大增加了他们的工作量。对每一种语言环境，他们都必须测试应用程序的每个功能是否运行正确。通常，要彻底地测试一个应用程序所需要的时间总是不够，更不用说对十几种语言了。而且测试人员往往对所测试的语言环境并不熟悉，有些对以这种语言为母语的人来说很明显的错误都有可能被忽略掉。

3 解决方案

针对上述三个环节中经常出现的问题，我们分别给出以下一些解决方法。

3.1 编程环节

在编写 I18N 程序时，程序员难以得到实际的语言字符串资源来调试代码。在设计中经常出现的一个错误是没有分配足够的缓冲区来存放字符串，在用户界面中没有足够的地方来显示字符串。这是因为从一种语言到另外一种语言的翻译往往会导致字符串的扩张，这种扩张有时甚至是成倍的，字符串的实际长度要直到运行时才能确定，所以在某些语言环境中就可能出现问题。另外一个问题就是双字节编码的支持问题，西文都是采用单个字节来表示字符的，而中文、日文等远东语言由于有大字符集，从而需要两个字节来表示一个字符，如果在设计时未考虑到对双字节编码的支持，那么在文字解码时就会出现问题，有时甚至导致应用程序的崩溃。

为了让程序员不需要等待实际的翻译后的语言字符串资源，在开发初期就能方便地调试国际化代码，我们采用了一种虚拟语言机制。通过人工构造一些"语

言”，每种“语言”的字符串主要模仿实际字符串的某一种特征，然后利用这些“语言”来测试应用程序，就能够得到实际“语言”环境下可能产生的一些主要问题的反馈信息了[3]。

例如，为了模拟产生长字符串的语言，我们就构造一种具有超长字符串的“语言”：将初始资源字符串都扩展三倍来形成新的“语言”资源。利用这种超长“语言”资源就能调试应用程序，看它是否能很好地支持字符串的扩展。又比如，为了模拟双字节编码字符串，我们将普通的 ASCII 码都映射到 UNICODE 码（每个字符用两个字节来表示，其中高位字节为0，低位字节为原来的 ASCII 码），这样利用它就能有效地调试应用程序中对双字节编码的支持问题了。

这种虚拟语言模拟机制的实现非常简单，只需要一个小时左右就能编出一些很小的程序来产生要“翻译”的语言资源。随着需要还可以产生各种各样具有不同特征的“语言”。程序员能够在编程时随时对应用程序的各种国际化特征进行调试，而不需要等待翻译小组的翻译结果。

3.2 翻译环节

如第2.3节所述，在翻译环节翻译人员不可能对语言文化相关的资源的结构有彻底的了解，一是因为不能假设所有的翻译人员都很熟悉计算机系统及应用程序结构，二是因为信息资源结构的复杂性。翻译人员在翻译时不可避免地要导入一些错误，为了加快翻译进度和减少附加的错误我们给出以下两种方法。

3.2.1 *自动翻译工具* 我们设计了一种自动翻译软件，这个软件能够自动识别信息资源结构，并从中分解出每个需要翻译的信息字符串；通过一个友好的用户界面让翻译人员输入相应的翻译后字符串；最后将翻译后的信息字符串保存为新的语言信息资源。同时它还维护一个翻译信息数据库，将所有的翻译信息都储存在数据库中，从而提供了一种持续的机制让翻译人员可以共享以前的或别人的翻译信息。使用这种翻译工具保证了信息资源结构的完整性，明显地减少了翻译时导入的附加错误；同时由于用户界面易于使用，在翻译信息数据库的帮助下极大地提高了翻译速度。

3.2.2 *翻译检测工具* 我们另外设计了一些简单的工具来检测翻译信息的结构性特征。这类工具仅仅只是检查翻译后是否导入了某种信息结构上的错误，比如检测翻译前后资源文件中标识符的匹配，关键字的完整等。象前面提到的“Window”关键字的翻译，以及标识符从1到11的误改都能被检查出来。我们用这种工具对一个程序的翻译后资源进行检查，发现了50多个错误。这种检测不需要运行应用程序，从翻译人员那里得到语言资源后，马上就可以独立地进行这种检测，而且可以根据实际情况来定制检测程序，不断地对它进行补充，最后得到完备的检测工具。

3.3 测试环节

对一个程序的测试总是费时费力的，尤其是对几种语言逐一进行测试。当测试人员要测试某种功能时，首先要建立第一种需要测试的语言环境，然后运行程序，再比较程序的输出是否与所要求的字符串一致；接着进行第二种语言的测试；如此直到所有的语言都测试完毕。为了在各种语言运行环境下测试一个程序，不得不雇用更多的测试人员。这种测试方法不能很好地适应国际化软件设计的需要。

3.3.1 *抽象的测试方法* 为了解决这个问题，我们结合第3.1节所述的虚拟语言，采用了一种抽象的自动测试方法。在进行功能测试时，首先提供一种虚拟语言，这个语言就是将实际语言资源中的每个资源字符串都替换为对应的资源字符串的标识符而得到。这样在测试时，让应用程序使用这种定制的虚拟语言，利用应用程序使用标识符从信息资源中提取字符串的特点，在测试要求中不提供字符串，而提供字符串标识符，然后比较输出的字符串所表示的标识符数字与测试要求中的字符串标识符数字是否相同。

由于这种方法只需要比较数字的标识符，而数字的比较总是非常简单可靠的，对测试人员也没有什么语言上的要求；同时对于不同的语言来说，信息资源的标识符都是一样的，所以一次测试就可以保证应用程序对所有的语言版本都是正确的。当然前提是翻译后的信息资源没有任何结构上的错误，而且也没有翻译上的技术性错误。这种方法特别适合于一些基本的功能测试。

为了保证没有信息资源结构上的错误，可以采用第3.2.2节所讲的翻译检测工具。为了保证没有翻译技术性的错误，下面引入了利用互连网络技术的新的方法。

3.3.2 *互连网络技术的利用* 为了使应用程序能够真正地“本地化”，就必须保证应用程序中的各种语言资源都能符合该种语言文化习惯的要求，也就是说要保证翻译后的各种信息都要符合使用这种语言的用户习惯。通常为了实现这个要求，需要雇佣以这种语言为母语的人员或者委托当地别的合作公司来测试。现在由于互连网络技术的发展，人们在世界各地都能方便迅速地访问某个网络站点上的信息，从而也提供了一种新的方法来保证软件的本地化[4]。

在公司中测试人员设置需要的语言环境并运行需要测试的应用程序，将每一步产生的用户窗口都从屏幕上截取下来，将它储存为 GIF 或 JPEG 格式的图像；

最后当所有的用户窗口都获得后，利用一个工具将所有这些图像按照一定的顺序生成为一个 HTML 文档，并将该文档放到公司的互连网络站点上。公司中的测试人员只需要尽可能完整地提供应用程序的用户界面，并且将该 HTML 文档尽量做得易于访问，比如在文档中要提供"向前"、"向后"等按钮，使得可以随意遍历各个用户界面。

然后将该站点的地址通知翻译人员、选定的最终用户以及合作公司等相关人员，邀请他们访问该站点，并对用户界面做出评价，提出改进意见。这样做的好处是：他们不需要费力去设置运行环境来运行该应用程序；可以保证所进行的评测是完整的；以前，当翻译人员开始翻译整个信息资源时，每个信息字符串都是孤立的、静态的，现在提供了上下文环境后，就可以用一种联系的、动态的观点来更进一步考虑翻译结果是否符合"本地化"的要求。按他们的意见改进后，截取新的相关画面来更新站点上的 HTML 文档。

总　结　软件的国际化具有极大的优点，同时也是当前发展的方向，但只有将国际化和软件开发过程的其他方面结合起来才能获得高质量的软件。目前所用的开发模式还不能将编码、本地化和测试等很好地结合起来。我们在这方面做了一些探索，希望能有利于以后更进一步的研究。

通过改变开发和测试过程，开发一些开发和测试辅助工具，我们已很大程度地改变了国际化程序的开发和测试过程，主要体现在以下几个方面：(1)编程人员能够早期调试程序的国际化特征；(2)翻译人员能够更快速准确地提供翻译后的资源；(3)测试人员能够更快速完整地测试国际化应用程序。

在软件的生产过程中采用这些方法，节约了国际化软件开发和测试中所使用的资源，加快了开发进度，同时还提高了国际化软件的质量。

参 考 文 献

1 Richard ishida, et al. UI Localization Design Guild, FujiXerox Co., 1999. A1～7

2 Globalization services, Uniscape Inc., 1999

3 Robinson H Using Poka-Yoke Techniques for Early Defect Detection. In: Proc. of the Sixth Annual Conf. on Software Testing, Analysis and Review. 1997, 119～142

4 Chakrabarti S, Robinson H. Catching Bugs in the Web. Using the World Wide Web to Detect Software Localization Defects. In: Proc. of the Tenth Interl. Software Quality Week. 1997. 7A2

(上接第17页)

3　关于 Java 与 AI 语言的结合

毋容置疑，Java 是迄今为止最为出色的网络程序设计语言，它应是"网络时代"的主旋律，对 Java 技术的发展和应用是软件与国际接轨并同步发展的明智之举。虽然 Java 并非一个专门的 AI 语言，但从以上讨论可以看出，它具有许多适合 AI 应用的特性和功能，本身又是一个集多种现代程序设计技术的诸多优点于一身的 OO 语言，故作为一种 AI 语言使用未尝不可。有关 Java 对 AI 技术影响和促进以及潜在的贡献远非本文所能涵盖，文本可能挂一漏万，实为引玉之砖。然而，Java 必定并非专为 AI 设计，因而不能完全适应 AI 应用的需求。特别是 OO 风格缺乏内含的推理机制，在知识表示和知识推理方面存在与其它 OO 语言，如 Smalltalk 类似的局限性。笔者认为，以 Java 为基础，结合专门的 AI 语言技术，是改进当前 AI 语言及开发工具发展滞后，为 Internet 用户提供更好的语言和工具的切实可行的做法。我们自1997年以来将 Java 用于 AI 语言 Tuili[7,8]的实现[9]以及 Java 与 Tuili 语言级的结合研究，取得了一些进展，有关内容另文介绍，我们仍将继续这方面的工作。

参 考 文 献

1 易文韬，陈颖平. Java 手册. 科学出版社，龙门书局，1997

2 张东摩，李红兵. 人工智能研究动态与发展趋势一参加第十五届人工智能联合大会总结报告. 计算机科学，1998，25(2)

3 郑国梁，邵维忠. 齐心协力一1997年国际软件工程会议概况. 计算机科学，1998，25(2)

4 王克宏主编，郁欣，等编著. Java 语言编程技术. 清华大学出版社，1997

5 陆汝钤编著. 人工智能. 科学出版社，1989

6 陆汝钤，等著. 专家系统开发环境. 科学出版社，1994

7 高全泉. 微机上实现的逻辑推理语言 TUILI1.1. 计算机科学，1992，19(5)

8 Lu Ruqian. TUILI-A language for expert systems. In: Xu kongshi ed. Advances in Chinese Computer Soience(1), Singapore: World Scientific Publishing House, 1988. 99～114

9 高全泉. TUILI(推理)语言编译系统的 Java 实现. 计算机学报，1999，22(4)

10 徐国平主编，曲大成，等编著. Internet 简明教程. 清华大学出版社，1997

11 张铖. 网络时代的人工智能. 计算机世界，1997. 1. 6