基于 Monte Carlo 方法的软件移植风险测试与实际处理过程

徐 明,孙小兵,詹 敏

(扬州大学信息工程学院,江苏 扬州 225127)

[摘要] 针对软件移植过程中可能出现的风险,本文分析了各种应对策略以及实际处理过程. 按照软件移植的风险测试及处理顺序,文中首先给出了功能风险因素分析,作为预测软件移植是否成功的首要标志. 在完成了软件移植过程中对于产生的各种错误进行测试以及处理之后,深入探讨了 Monte Carlo 测试法,作为移植后软件正确性、稳定性的一种最终检测手段,凸显在软件移植全过程中的重要地位. 文章以 JSP 平台到. NET 平台转换为例,详细阐述了软件移植的具体过程. 该软件移植风险分析思路已成功运用于某管理系统项目的软件移植过程中.

[关键词] 软件移植,风险因素,蒙特·卡罗方法,JSP,. NET

[中图分类号]TP3 [文献标志码]A [文章编号]1001-4616(2014)04-0131-06

The Risk Test and Actual Process of Software Migration Based on Monte Carlo Method

Xu Ming, Sun Xiaobing, Zhan Min

(College of Information Engineering, Yangzhou University, Yangzhou 225127, China)

Abstract: This paper analyzed various strategies and practical process according to the possible risks in the process of software migration. In accordance with the sequential of the risk test and treatment, this paper first gave the functional analysis of risk factors, as the primary symbol of successful migration of prediction software. At the completion of the process for software migration after produces all kinds of error testing and treatment, this paper studied the Monte Carlo testing method in-depth as the final detection means for software correctness and stability of post migration, this method highlights the important position in the whole process of software migration. This paper took the translation from JSP platform to. NET platform. This example had successfully applied in a practical management system project for the software migration.

Key words: software migration, risk factor, Monte Carlo method, JSP,. NET

在软件项目开发过程中,由于系统自身硬件环境、软件运行环境、不同软件平台的特性、以及用户特定的应用要求等诸多环境因素,常需要将某个软件系统从一种开发平台移植到另外一种开发平台中. 由于不同软件开发平台的自身功能性特点,在某一个平台开发的软件系统,其部分功能往往不能很好地移植到另一软件平台上,甚至无法实现其功能. 同时,由于平台或环境不同,而造成大量错误的产生,最终导致无法实现软件移植工作. 因此,若要成功实现一个软件系统的跨平台移植,必须考虑其潜在的移植风险[1]. 移植过程可能是成功的,也许最终会失败. 在进行大规模、系统性移植之前,必须制订周密的软件移植方案,可先期进行一些部分单元或特定功能测试,以确保整个移植工作的成功[2]. 在消除各种错误、实现所有功能后,还必须测试移植后软件的可靠性,因此,选择合适的测试用例,以便覆盖所有测试路径,是确保测试能够完全成功,从而降低移植风险的一个重要步骤.

软件移植通常需要考虑如下一些风险因素:首先,原有的软件功能移植到其他软件开发平台上是否能

收稿日期:2014-08-16.

基金项目:2012 国家级大学生实践创新训练计划立项项目(2012JSSPITP1332).

通讯联系人:徐明,讲师,研究方向:软件测试. E-mail:xuming@ yzu. edu. cn

够全部实现;其次,移植过程中出现的错误如何处理,包括对于泄漏错误、移植过程中产生的新错误、软件平台不兼容错误等相关错误的预防与处理;最终,还要选择相应的用例,检测移植后软件的稳定性、正确性,以确保软件移植工作是成功的.

1 基于软件功能的移植风险因素分析

对于软件项目开发人员而言,用户需求能否转化为具体功能,最终功能能否全部实现,是衡量软件开发是否成功的重要标志. 因而,软件移植首先考虑的最优结果是:原软件环境下的所有功能在新的软件平台上毫无折扣地全部实现. 即,软件移植成功的标准为软件功能全部实现. 此时,风险因子为0,为最理想的目标. 然而,实际移植过程中,由于不同的软件环境、对于项目本身缺乏深入了解以及不同的软件开发人员的软件技能差异,导致实现用户所需功能的可能性风险不断增加(R_p);此外,移植过程中成本的花费、软件质量的下降等风险性影响因素不断加大(R_i). 最终导致的风险因素可用如下公式表示[3]:

$$R_f = R_p * R_i.$$

式中:R_f 为 Risk Factor,表示风险因子;R_p 为 Risk Probability,表示风险可能性;R_i 为 Risk Impact,表示风险影响.

这里 R_p 取值范围为0~1,分别表示风险发生可能性的由低至高,R_i 取值范围为1~10,数字表示风险影响的大小,软件移植全面实施之前,必须给出正确的评估、测试,可采用取样分析的方法,取部分独立单元代码先期进行移植、转换,采用软件测试的方法给出 R_f 最终结果. 若该数据接近0,表示可以顺利实施软件移植. 相反,则表示风险较大,如果数据接近10,则表示该软件项目根本不能进行软件移植,开发人员需要在新的软件平台下重新编写所有代码.

针对具体的软件项目为例,我们首先对于某登录模块 shenbao/index.jsp,以及显示模块 s_showInfo.jsp 进行移植测试,并给出评估结果,其风险因子为0.23. 通过测试实例,证明本软件项目经过移植后仍然可以很好地实现原系统的功能,可以进行全面软件移植. 这里转换后的具体.NET平台代码省略.

2 软件移植过程中错误分析与处理过程

2.1 综述

我们经过测试,首先可以确保软件移植后系统功能完全实现,从而减少了软件移植的功能风险. 但是在软件移植具体过程中,不可避免地会产生各种错误,这些错误,同样是构成软件移植的重要风险因素. 本文以某单位《基于.NET的大学生创新项目管理系统》为背景,描述软件移植风险分析,及移植过程中的软件测试处理过程. 原系统采用JSP环境开发,根据用户要求,将JSP开发环境的软件系统移植到.NET环境. 因此,本节中所给出的实例,均取材于项目开发团队在代码编写、软件移植所遇到的各种问题,以及最终的解决方案.

2.2 泄漏错误的分析与处理

泄漏错误是指在软件开发生命周期从某个开发阶段带入到下一个开发阶段的错误[4]. 这里指软件移植前的错误在软件移植后,被带入新的环境中,仍然没有解决的错误. 软件移植通常是由于现有运行环境改变,或者用户提出要求开发的软件系统在其它软件环境下运行而产生的新任务. 软件移植不可能在软件开发、调试完全结束之后再进行. 一旦需要改变运行平台,要尽早实施软件移植. 例如,由于.NET支持HTML描述,通过在.aspx中嵌入java语言标记<script language=JavaScript></script>将JSP平台的java语言源代码移植到.NET中运行[5]. 此时,通常原系统中的错误并没有解决,错误被带入新的运行平台后,仍然存在. 运行界面,及相关代码如图1所示.

这是邮件发送截图. 在软件移植之前虽然采用使用javaMail技术,由于调试始终存在错误,在JSP环境下一直没有完成实现发送邮件的功能(代码略). 然而在.NET环境中却很容易实现邮件发送功能. 因此,我们进行了重新设计,下面是在.NET环境下实现该邮件发送的相关代码:

```
try{
        out.println("Hailing...");
        email.hail(from,to);
```

```
                out. println( email. getServerReply( ) ) ;
                out. println( " <br>" ) ;
                try{
                        out. println( " Sending Message. . . " ) ;
                        email. sendMessage( from, to, subject, message) ;
                        out. println( email. getServerReply( ) ) ;
                        out. println( " <br>" ) ;
                    } catch( SMTPException e) {
                         out. println( e. getMessage( ) +" <br>" ) ;
                    }
              try{
                        out. println( " Logging off. . . " ) ;
                        email. logout( ) ;
                        out. println( email. getServerReply( ) ) ;
                        out. println( " <br>" ) ;
                       } catch( SMTPException e) {
                out. println( e. getMessage( ) +" <br>" ) ;
                }
        } catch( SMTPException e) {
        out. println( e. getMessage( ) +" <br>" ) ;
        }
    } catch( SMTPException e) {
        out. println( e. getMessage( ) +" <br>" ) ;
        }
```

图 1　邮件发送界道

Fig. 1　The interface of mail sending

2.3　新错误的分析与处理

新错误是指在软件开发生命周期某个开发阶段新产生的错误[4]. 这里指经过软件移植之后产生的新错误. 比如说,虽然可以将 java 语言嵌入到. aspx 中运行,这只是对于简单的 java 语句,由于一个系统中包含许多函数以及标准类库等,在新的. NET 平台下运行会存在许多的不兼容性甚至错误等事先无法预料的问题,这些不确定因素导致大量新错误的产生,因而也是必须引起重视的软件移植的重要风险因素之一.

由于平台转换而带来的新变化,肯定会有新的错误产生,这是不言而喻,也是我们在软件移植过程中最容易料想到的问题. 限于篇幅,对应于本软件项目实例中,关于运行界面的转变以及对应代码的更新,这里不逐一赘述.

2.4　兼容性错误的分析与处理

兼容性错误是指不同模块各自独立运行时正常,当把他们集成到一起时,产生的相互不兼容的错误[2]. 对于软件移植而言,兼容性错误应该是指由于系统移植到不同的平台造成的不能或者无法产生原有的功能效果,此时,只能采取相应的替代方式,用新软件开发平台所提供的其它工具,实现系统对应的功能要求. 对应于本项目而言,运行界面,如图 2 所示.

图 2　专家信息编辑界面

Fig. 2　The interface of specialist information editing

该部分功能从 JSP 环境移植到. NET 之后,由于 select 控件在. NET 环境下不兼容,表面上看,系统没有错误可以正常运行,但点击“添加”按钮在右边文本框中得不到所需添加的相应选项内容,主要是由于在 JSP 下,select 控件以数组形式命名,调用 javascript 函数时以实参形式传递该数组名没有任何问题. 转换至. NET 平台下,如上调用则出现控件接收不到数据的问题. 因此需要把 select 控件修改为两个 input(text)控件用以接收数据. 将 JSP 环境移植到. NET 环境后,通过替代方式,实现对应功能的转换[6,7]. 此处代码部分略.

2.5　软件移植中的其他风险因素

在软件移植中还需要考虑的相关问题或者风险因素有:软件移植系统功能前后一致性问题,软件移植复杂性以及成本核算,软件移植后系统可维护性问题,软件移植完成后系统的通用性和可扩展性问题等. 限于篇幅,这里不展开详述.

3　基于 Monte Carlo 的软件可靠性测试

3.1　Monte Carlo 方法基本思想

软件移植中的错误是我们大量面临而且必须首要解决的问题,减少和预测错误的发生,将极大程度地降低移植风险,否则将无法完成移植任务. 但在软件移植过程中,仅仅依靠发现和预判错误的发生,还远远不够. 我们还必须对移植后的程序代码,进行可靠性测试分析,以确保代码移植后系统的稳定性和正确性. 动态白盒测试是实现对于软件整体结构测试的一种常用方法,也是完成软件测试覆盖技术的常用手段[8]. 众所周知,测试覆盖率不高或者不够全面,这是白盒测试乃至传统测试等所无法克服的致命缺陷. 必须改进测试手段,以确保对于软件移植后代码的可靠性、稳定性测试.

Monte Carlo(蒙特·卡罗)是一种随机模拟、随机抽样测试法,被广泛用于以统计模拟为数学模型的随机模拟应用系统中[9]. 实践证明,采用蒙特·卡罗测试法,能将白盒测试与黑盒测试有效结合起来,通过随机产生多组测试用例,能够达到对于测试路径的完全覆盖目的[10],尤其是在比较型测试中,将极大地提高软件测试的实用性和有效性.

采用蒙特·卡罗测试法,首先我们必须选择一个随机变量 x,然后计算该变量的简单子样算术平均值[11]:

$$\overline{x_N}=\frac{1}{N}\sum_{n}^{N}x_n\ ^{[5]},$$

这里,$\overline{x_N}$为 I 的近似值,根据中心极限定理,对于任何 $\lambda>0$ 有:

$$P(|\overline{x_N}-I|<\frac{\lambda_\alpha\sigma}{\sqrt{N}})\approx\frac{2}{\sqrt{2\pi}}\int_0^{\lambda_\alpha}\mathrm{e}^{-0.5t^2}\mathrm{d}t=1-\alpha^{[5]},$$

如果 $\sigma\neq0$,那么蒙特·卡罗方法的误差为:

$$\varepsilon=\frac{\lambda_a\sigma}{\sqrt{N}}\ ^{[5]}.$$

3.2　Monte Carlo 在软件测试中的具体实现

首先设定评判准则:设 $e_1,e_2,\cdots,e_k$ 是所有 k 个基于逻辑覆盖和基本测试方法设计生成的测试用例,$E=\{e_1,e_2,\cdots,e_n\}$,$P(\{e_1\})$表示 t 时刻第 i 个测试用例发生的概率,则有:

$$P(\{e_1\})=P(\{e_2\})=\cdots=P(\{e_k\})=1/k^{[5]},$$

$$P(\{e_1\})=P(\{e_2\})=\cdots=P(\{e_k\})=1^{[5]}.$$

这里要求各测试用例必须具备完全独立性和完全覆盖性,被测试软件是完全正确的.

按照上述要求,我们选定测试用例. 下面的程序来自《基于. NET 的大学生创新项目管理系统》,用于判定用户登录系统是否成功:

```
if(p1==Session["p_id1"]&& pw1==Session["p_pw1"])//条件 1
        {
            Session["p_loginSign"]="OK";
            Response.Redirect("p_index.aspx");//程序段 1 按账号登录成功
        }
if(p2==Session["p_id2"]&& pw2==Session["p_pw2"])//条件 2
        {
            Session["p_loginSign"]="OK";
            Response.Redirect("p_index.aspx");//程序段 2 按身份证号登录成功
        }
rs.Close();
conn.Close();//程序段 3 关闭连接
```

根据上述程序代码,建立条件判定流程图,标记程序的执行路径:

根据可能出现的条件,程序运行将选择不同的路径,如:p1==Session["p_id1"]为 T,pw1==Session["p_pw1"]为 T,p2==Session["p_id2"]为 T,pw2==Session["p_pw2"]为 T,选择路径 abd,覆盖分支 bd;p1==Session["p_id1"]为 F,pw1==Session["p_pw1"]为 F,p2==Session["p_id2"]为 F,pw2==Session["p_pw2"]为 F,选择路径 ace,覆盖分支 ce;….

最终结果:分析测试用例,由于各测试是独立,测试结果概率值均达到 1,说明被测试程序具有完全覆盖性;再对测试值和预期结果比较,本测试中得到的值和预测结果相同,说明满足评判准则,被测试程序通过测试.

限于篇幅,对应于本测试用例的具体列表及计算过程,在此不逐一赘述.

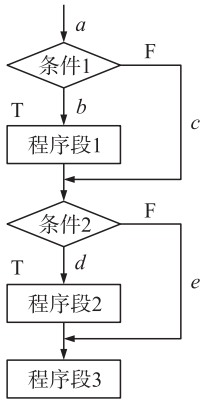

图 3　登录判断流程图

Fig. 3　The flow chart of login decision

4 结论

在软件移植过程中,不断面临一次次新的风险、新的挑战,常常使我们修改、放弃原来的方案,重新寻找新的途径.有时也会让我们退却,甚至萌发重新编写整个系统的念头.因此,对于一个大型软件系统,在大规模实施软件移植之前,必须对于软件移植风险作出周密、综合的评判及分析,明确、细致地列出可能影响软件移植的各种风险因素,并加以处理.判断软件移植是否成功,最终用例选择以及测试至关重要.成功的软件移植,不仅会给原来的代码增加不一样的色彩,也会让我们对新、旧平台的区别和联系有更深刻的认识.

本文所述的风险分析、测试,已经成功运用于实际的《基于.NET的大学生创新项目管理系统》,本项目初期采用JSP开发,根据用户要求,最终成功移植到ASP.NET环境,系统现在处于试运行阶段,目前正在进一步完善、丰富系统的功能.

[参考文献]

[1] 黄聪会,陈靖,张黎,等.软件移植理论与技术研究[J].计算机应用研究,2012,29(6):2 024-2 027.

[2] Mariano Ceccato, Thomas Roy Dean, Paolo Tonella, et al. Migrating legacy data structures based on variable overlay to Java[J]. Journal of Software Maintenance and Evolution Research and Practics, 2010, 22(1): 211-237.

[3] Roger S. Pressman. Software Engineering: A Practitioner's Approach[M]. 7th edition. Columbus: McGraw - Hill, 2010.

[4] William E Perry. Effective Methods for Software Testing[M]. 3rd edition. Hoboken: John Wiley & Sons Inc, 2006.

[5] Yang Qiuxiang. Application research on software reuse technology: image and signal processing (CISP) [C]//2010 3rd International Congress, Yantai, 2010.

[6] Alok Ranjan, Rajeev Kumar, and Joydip Dhar. A comparative study between dynamic web scripting languages[C]//2nd International Conference on Data Engineering and Management, Tirachiraphalli, 2010.

[7] Chouki Tibermacine, Mohamed Lamine Kerdoudi. Migrating component-based Web applications to Web services: towards considering a "Web Interface as a service": Web services (ICWS) [C]//2012 IEEE 19th International Conference, Honolulu, 2012.

[8] Yang Shunkun, Zeng Fuping. Improved genetic algorithms for software testing cases generation[C]//2012 4th Electronic System-Integration Technology Conference, Ansterdaun, 2012.

[9] 马海云,魏凯斌.一种新的软件测试方法的研究[J].自动化与仪表,2010(3):4-5.

[10] Li Qiuying, Ji Ping. Study on the determination method of the minimal software reliability test effort based on sampling probability[C]//Proceedings of 2011 3rd International Conference on Computer Research and Development. Shanghai, 2011.

[11] 马海云,张少刚.软件质量保证与软件测试技术[M].北京:国防工业出版社,2011.

[责任编辑:陆炳新]