日本软件质量保证实践

谭守标,徐 超,李正平
(安徽大学 电子学院,安徽 合肥 230039)

摘 要:日本在软件项目开发上强调质量第一。主观上,诚信及服务型社会的氛围促使每个开发人员自觉提高质量;客观上,通过采用合理的项目开发过程、配备质量管理人员、开发中及时寻求技术支援、基于概念清晰的软件架构进行开发、利用多种测试手段充分测试、重视文档等手段来进行质量控制。

关键词:软件工程;质量保证;软件架构;测试;文档
中图分类号:TP311.52 **文献标识码**:A **文章编号**:1673-629X(2007)06-0046-03

Practice of Software Quality Assurance in Japan

TAN Shou-biao, XU Chao, LI Zheng-ping
(School of Electronic Science and Technology, Anhui University, Hefei 230039, China)

Abstract: Quality is the most important thing in software project in Japan. The atmosphere of the integral and service-oriented society impels every developer improving software quality consciously. In addition, software quality is controlled by a lot of means, such as adopting feasible development process, equipping with quality management staff, seeking technical support actively, developing based on a framework with clear concepts, adequately testing by kinds of methods, paying great attention to documents, etc.

Key words: software engineering; quality assurance; framework; test; document

半年来在日本参与一个大型公司的软件项目开发,对比多年在国内开发日本项目的经历,感到日本软件项目开发方面有很多经验值得借鉴,试从以下几个方面谈谈感想。

1 质量理念

质量和成本,是衡量一个项目成功与否的两个关键因素。成本可以在项目间消化,而质量却直接关乎到一个项目的成败;另外,通过对质量的早期控制,也能降低整个项目的开发成本。因而,对一个项目来说,质量是最为重要的。

软件质量保证不仅追求"零"错误率,更应该使软件给客户带来实际效益。

对软件企业而言,软件质量甚至是攸关企业生命。软件中一个很小的失误,都有可能给客户造成巨大的损失,而这个损失又会由软件开发企业自己来承担,因而可能会直接威胁到企业的生存。如日本东证公司的证券交易软件因为没有完善的取消功能,当某一证券公司将"1股60万的股票,出售1万股",错输成"1股1万,出售60万股"后不能正常取消而损失惨重;又如日本铁道公司(JR)订票系统,因一个小的错误(Bug),使得某一天无法订票,导致数万人出行困难。

软件质量问题,往往被认为是看不见、摸不着的事情,很难准确去把握。实际上,现在的观点是,软件质量和普通产品的质量一样,是可以用定量指标去衡量的,也是可以通过各种手段去保证的。

2 质量保证原则

质量要从项目一开始就进行控制。质量越早控制,项目整体质量就越高[1,2],成本也能降得越低。质量控制的越早,开发过程中出现反复的可能性也就越低,潜在的隐患也就会越少。而问题发现的越晚,付出的代价就会越大。

宁慢勿错,宁用笨办法,也不用危险的方法。动作快虽然能够提高效率,但是也容易出错。而一旦出错,就会使开发过程出现反复,反而影响效率。在日本能明显感觉到这种怕犯错误的无形压力,做什么事情都

收稿日期:2006-09-17
基金项目:对华日元贷款资助项目(中国内陆地区人才培养项目)
作者简介:谭守标(1976-),男,湖北天门人,副教授,博士,研究方向为计算机仿真教学、软件质量保证;徐 超(1962-),男,安徽亳州人,教授,博士,研究方向为信息系统规划、网络计算;李正平(1978-),男,安徽宣城人,副教授,博士,研究方向为计算机应用,软件工程。

要反复确认。

3　质量保证手段

从主观和客观上两方面着手,进行有效的质量控制。

主观上,诚信社会的氛围,使得每个开发人员都自觉地去保证质量。自己承诺做过的事情,就应该保证在那一点上没有错误,如果出现错误,特别是多次重复,就会使自己失去诚信,所以出现一次错误,就会战战兢兢的,会不断努力,去避免错误。另外,日本社会的"客户就是上帝"的良好服务意识,也使得每个人都具有较高的质量意识。不只是考虑如何实现功能,同时会考虑怎么样去提高可用性、可靠性,及实现产品的人性化。

客观上,从开发过程、资源构成、软件架构、测试手段、文档等方面来进行质量控制。

3.1　项目开发过程

多采用瀑布模型[3],包括项目计划、外部设计、详细设计、测试用例设计、开发、单体测试(UT)、结合测试(CT)、系统测试(ST)、发布。

值得一提的有:

在外部设计阶段,常驻客户处了解客户业务及需求,先做好画面原型,帮助客户明确需求,方便地将式样确定下来,尽量避免后期的式样变更和追加。同时力争利用软件改善客户业务流程,往往一个项目下来,软件人员能比客户业务人员更理解客户业务。

通常的做法是先开发出代码,再进行测试用例设计和测试。更新的观念是:先应该设计好测试用例,再指导代码开发。这样更能促使代码的整体结构优化。

3.2　项目资源构成

一个大的项目,往往需要很多人力资源,而一个公司通常无法在每个时间段都能保证这么多的人力资源,但不会以不足的人力而牺牲项目质量。因此,很多软件公司之间都存在一定的合作关系,很多项目都是由一家公司分包给多家软件公司合作完成。项目过程由发包公司统一管理。

项目的另外一种人力资源是转勤人员。当项目规模不足以需要分包时,则会从合作公司借调员工,由公司统一管理,使得公司能以较低的成本高质量完成项目。因此,转勤人员在日本大量存在。

为进一步降低成本、提高质量,很多项目选择外包方式,即将项目的一部分业务或者部分过程发包给劳动力相对便宜的国家,如中国、印度。这样能充分利用国外的优秀资源,在降低成本的同时提高质量。由于国家之间语言不同,地域相差较远,因此外包项目需要特殊的人力资源,整体资源图如图1所示。

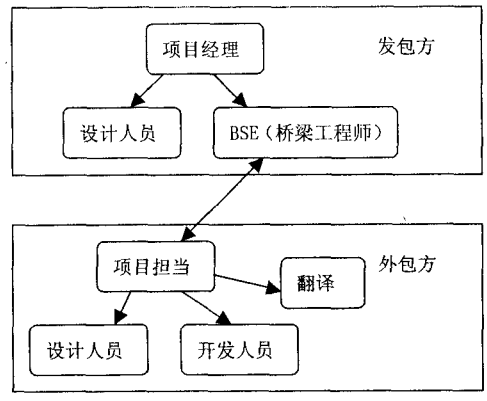

图1　外包项目人力资源图

项目组内,除了常规的开发人员外,还有专业质量管理人员,通过各种文档去分析各种规律,及时发现潜在问题,帮助项目组提高质量[4]。除此外,还会预留人手,确保项目按质保量完成。比如,项目组因为有我的参与,所以专门配了翻译。虽然后来翻译大多时候并不需要,但为保证式样的准确把握和开发过程中的顺畅交流,翻译一直保留。

3.3　寻求技术支持

"术业有专攻",项目在开发过程中如果碰到什么疑难问题,会积极寻求外部的技术支持。如碰到数据库性能方面的问题,会直接寻求数据库软件公司的专业技术人员,这样就会在较短的时间内寻求到很好的解决办法。

3.4　软件架构

一个好的体系结构,是一个项目成功的基石。日本大多项目均建构在如 Struts 等优秀的底层上,并根据项目特点,在中间件基础上再扩展出一套公共组件,如著名的 OME,NSF 等。这样既使得项目整体架构合理、概念清晰,减少发生错误的机率,也使得开发过程省力不少。

3.5　各种测试手段

后期保证质量的直接方法就是进行各种检查和测试,消除前面遗留的错误。主要分为静态检查、单体测试(UT)、结合测试(CT)、系统测试(ST)。静态检查主要是利用各种工具,通过静态分析检查出程序中不合理语句和编程方法[5]。UT 包括 JUnit 测试[6,7]、画面测试[8]、代码覆盖率检查[9]等。

为了最大化测试效果,构建合适的测试环境相当重要,如开发/UT 环境,CT 环境,ST 环境,实际运行环境。实践证明,很多问题常常会暴露在特定的环境中。

3.6　重视文档

文档既是交流的重要手段[10],同时也是留存的证

据,可以推进工作的流转,用于查找问题的根源,还可以进行各种统计分析工作[11,12]。文档使开发者不能掉以轻心,通过写文档也能让开发者发现工作中的问题,如程序修正票(P 票)就能使程序修正者再次确认修正的正确性。

日本项目的文档之多让人深有体会,每个公司也都有自己独特的文档格式,试列举部分管理文档如下:QA 票、悬案管理票、进度管理文档、Bug 票(B 票)/问题管理票(M 票)/式样变更票(C 票)、程序修正票(P 票)、UT 测试文档(PCL)、CT 测试文档(CCL)、源码依赖票、品质管理票。

3.7 控制纳品后的错误修改过程

当程序经过测试并纳品后,对程序的修改将被严格控制起来。当实际运行中再发现错误需要修改时,流程如图 2 所示。首先由开发方提交程序修正申请,填写修改的原因、修正对象、方案、影响范围、需要的时间等。申请由项目负责人和客户一起讨论并批准后,将要修正的程序返回给开发人员进行修改,修改完成后需要对牵涉到的地方进行严格的再测试,填写程序修正票(P 票)、UT 测试文档(PCL)、CT 测试文档(CCL)等文档。填写的文档进行审查确认无误后,项目负责人与客户商谈合适的发布时间再进行重新发布,确保尽量减少损失。

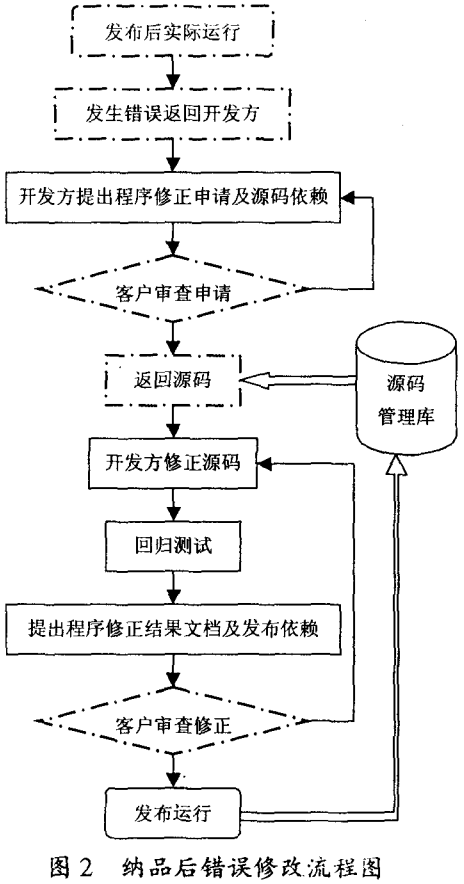

图 2 纳品后错误修改流程图

3.8 加强竞争

几个组同时开发,逐渐挑选出质量好的组进行后续开发。

4 存在不足

尽管日本软件业在项目开发过程中有许多值得借鉴的地方,但也存在着一些不足。如:过于繁琐、不够简练;过于迂腐、不够灵活。修改一个很小的问题,也需要经过十来道手续,费时两三天。虽然小心谨慎可以避免很多错误,但任何事情都应该有个度,过于小心谨慎后反而会使工期变长、成本成倍增加。

日本项目开发常常会分成几个组来开发,虽然能各尽所能、分散风险,但沟通上需要花费不少代价,且会存在很多隐患。底层、数据库的设计强调了可扩展性,但使得性能上出现了问题。

5 结束语

软件质量保证的理论往往是在实践中积累、总结的,经验十分重要。日本软件行业在长期的项目开发过程中积累了丰富的经验,本文仅是半年进修期间的个人感受,希望能让软件研发工作者有所借鉴。我国要增强软件行业的国际竞争力,成为软件出口强国,应多与欧美、日本等国进行软件项目合作,从中学习他们的理念和实践经验。

参考文献:

[1] Mizuno O, Kikuno T, Inagaki K, et al. Analyzing effects of cost estimation accuracy on quality and productivity[C]// Proceedings of the 1998 20th International Conference on Software Engineering. Kyoto, Japan: IEEE Comp Soc, 1998: 410-419.

[2] 钟剑龙. 试论软件产品的质量控制[J]. 科技进步与对策, 2003(4):84-86.

[3] 赵红波. 软件产品的质量控制[J]. 航空标准化与质量, 1998(2):37-40.

[4] Tanaka T, Aizawa M, Ogasawara H, et al. Software quality analysis & measurement service activity in the company[C]// Proceedings of the 1998 20th International Conference on Software Engineering. Kyoto, Japan: IEEE Comp Soc, 1998: 426-429.

[5] Ogasawara H, Aizawa M, Yamada A. Experiences with program static analysis[C]//Proceedings of the 1998 5th International Software Metrics Symposium. Bethesda, MD, USA: IEEE Comp Soc, 1998:109-112.

[6] Takahashi J, Kakuda Y. Testing for High Assurance System by FSM[J]. IEICE Transactions on Information and Systems

(下转第 52 页)

甲在整个行程中经历了从出发到和汽车乙在途中相遇的过程,把它划分为一个阶段,出发表示一个阶段的开始,相遇表示一个阶段的结束。同理,汽车乙也可以分成一个阶段。由于行程中的隐含信息可以由副词产生,也可以由事件产生,通过对主体模板来挖掘隐含的信息。

根据出发和相遇的两个事件,可以描述出汽车甲的主体模板:

(主体(主体号 1)(阶段号 1)(起始时刻 StartTime11)(终止时刻 EndTime11)(起始地点 A)(终止地点 C))

同理,可得到汽车乙的主体模板:(主体(主体号 2)(阶段号 1)(起始时刻 StartTime11)(终止时刻 EndTime11)(起始地点 B)(终止地点 C))

上面的两个主体模板就把隐含信息挖掘后的结果给出,由于两个主体的出发事件是"同时"发生的,所以两个主体的阶段起始时刻槽设置成相同;由于"相遇"事件的发生,两个主体的阶段终止时刻槽和阶段终止地点槽相同。

如果行程主体之间只进行孤立的处理,这样得到的结果是不全面的,必须处理不同行程主体之间的相互关系。通过对主体模板的比较分析,可以得到行程关系模板:

(行程关系 (速度关系 空)(时间关系 0)(路程关系 0)(阶段号 1_ *1)(主体号 1_ *2))

这个行程关系模板表示主体 1 的第 1 阶段和主体 2 的第 1 阶段进行比较,速度关系槽中的空表示速度上不能得到结果,时间关系和路程关系槽中的 0 表示两个主体阶段的差值为 0,也就是主体 1 第 1 阶段和主体 2 的第 1 阶段在速度上和在运行的路程上都是相等的。根据给出的行程关系,然后通过系统建模,就能得到这个行程问题的表达式,T11 = T21,S11 = S21。这样就挖掘出了行程中的隐含信息:两个运动主体的运动时间和运动路程是相同的。通过这样的流程处理就比较完整地理解了行程问题中的信息。

6 结束语

文中介绍了自然语言研究中语用学的一些理论,着重提到了认知语用学的一些研究成果。并且结合教学智能辅助系统,讨论了在语用研究中很重要的两个方面的内容:行程领域语境假设和行程领域的语用推理。最后给出了语义处理结果如何与具体语境结合产生语境效果,再根据行程领域的语境特征运用语用学理论实现行程领域中的语用推理,推导出一段自然语言在行程领域中的隐含信息,然后以数学形式给出全部信息。本系统要实现语用理解必须要建立一个关于应用领域的描述,即给出语境假设[2],所以一个领域的理解程度和该领域的描述状况对语用的理解、对隐含信息的挖掘具有相当重要的影响。建立合理完整的语境假设,才能够理解好自然语言。对于其他领域的语用实现,只需要建立一个属于该领域的完整的静态问题模板,并且和语义结果结合,就能推理出隐含信息。文中的系统留出了一些接口,便于领域扩展。

参考文献:

[1] Allen J. 自然语言理解[M]. 第 2 版. 北京:电子工业出版社,2005.

[2] 孙红梅. 认知语境和语言的理解[J]. 宜宾学院学报,2002,4(13):43-45.

[3] Jurafsky D, Martin J H. 自然语言处理综合[M]. 北京:电子工业出版社,2005.

[4] 王 微. 论认知语用推理中的传统语境因素[J]. 天津外国语学院学报,2005,12(2):44-46.

[5] Sperber D, Wilson D. Relevance: communication and Congnition[M]. 2nd edition, Oxford: blackWell, 1995.

[6] 刘根辉,李德华. 计算语用研究的方法和途径[J]. 计算机工程应用,2005,25(1):4-8.

(上接第 48 页)

tems, 2003, E86-D(10): 2114-2120.

[7] Nomoto H, Katahira M, Fukatsu T, et al. Independent test [C]//Proceedings of the First IAASS Conference - Space Safety, a New Beginning. Nice, France: European Space Agency, 2005: 501-506.

[8] Yamaura T. How to design practical test cases[J]. IEEE Software, 1998, 15(6): 30-36.

[9] Indue S, Yamada S. Testing-coverage dependent software reliability growth modeling[J]. International Journal of Reliability, Quality and Safety Engineering, 2004, 11(4): 303-312.

[10] Nakanishi T, Saeki M. Applying multiple program graphs to modify specifications[J]. IEICE Transactions on Information and Systems, 2000, E83-D(4): 669-678.

[11] Paul R, Ohara S, Tsunoda F, et al. A software test and evaluation environment based on longitudinal database[J]. International Journal of Software Engineering and Knowledge Engineering, 2002, 12(3): 223-244.

[12] Sawada K, Sandoh H. Software reliability demonstration testing with consideration of damage size of software failures[J]. Electronics & Communications in Japan, Part III: Fundamental Electronic Science, 1999, 82(5): 10-21.