

软件进化研究综述

惠长江, 吴 江

(西北大学 信息科学与技术学院, 陕西 西安 710127)

摘 要: 软件进化是软件产品交付给客户之后所发生的一系列改进活动, 是有目的地从早期版本来产生新版本的过程, 是软件工程中的一个重要领域。对软件进化进行了综述性的介绍, 讨论了其定义、发展历程及主要研究内容, 介绍了指导软件进化的几种典型方法, 分析了人们在软件进化过程中面临的一些挑战和问题。

关键词: 软件工程; 软件进化方法; 元对象协议

中图分类号: TP311.5

文献标识码: A

文章编号: 1673-629X(2007)04-0196-05

Survey of Software Evolution

HUI Chang-jiang, WU Jiang

(College of Information Science and Technology, Northwest University, Xi'an 710127, China)

Abstract: Software evolution is a set of activities which improve software once it is delivered to a customer, it is a process which generates a new software version from an earlier version, and it is an important sub-area of software engineering. The authors give a survey to software evolution. Firstly, describe the definition, development history and the main research contents of software evolution. Then describe some software evolution methods. Lastly the paper discusses some challenges in software evolution.

Key words: software engineering; software evolution method; meta object protocol

0 引言

随着现代软件功能越来越强大, 其结构也变得十分复杂, 加之常常为了获得一些额外的功能且要容易实现, 人们就把一部分由硬件实现的功能转由软件实现, 这样大大增加了软件的复杂度, 又因软件要受到各方面不断变化的驱动(如客户要求的改变及外部环境的改变等), 使得没有一个稍具规模的软件能够实现一次性的完全开发, 每个软件系统在投入使用后都要不断地改进和完善。对许多软件的一系列调查表明, 软件进化和维护的费用占到整个软件生命周期总费用的40%到90%之间^[1]。因此, 软件进化一直是软件工程研究的热点之一。

目前, 国内对软件进化的研究工作还偏少, 还很少把软件进化的思想融入到传统的软件开发过程中, 对软件进化还没有较好地进行系统研究。笔者对软件进化的相关方面进行了简要的介绍。

1 进化的定义

1.1 进化的一般定义

进化一词描述了许多不同领域中(包括抽象和具体的)值得人们关注的一类现象^[2], 它涉及到了一系列的实体, 如自然界物种、社会、概念、理论、思想等。如果它们的某些属性随时间进行着不断前进的改变, 就可说它在进化。进化一般指的是在其领域中它们属性或特性的前进性改变。所谓前进性改变是指在某种意义上取得了进步, 更有利于其自身的生存发展, 而这种进步或许会导致新属性的出现。一般来说, 这种改变是由适应单个实体或整个种类的需要来驱动的, 目的是维护和提高它们在一个变化的环境中的适应性, 因而这种改变可以使其更加有用和有意义, 或在某种意义上可以增加它们的价值, 同时这种改变可以去除那些没有价值和已经不适用的属性。

1.2 软件进化的定义

进化可以从许多不同的观点和领域来解释和研究^[3], 在软件领域中, 它通过对软件系统从最初的概念到可操作实现的过程研究, 通过对软件系统为了更好地适应外部环境而进行相应的进化和扩展的研究, 来研究软件进化的方法和意义。进化在软件领域的观点主要集中在进化的机制和工具上。

收稿日期: 2006-06-22

基金项目: 陕西省科技攻关项目(2003k05-g32)

作者简介: 惠长江(1981-), 男, 陕西延安人, 硕士研究生, 研究方向为语义网技术、E-learning; 吴 江, 副教授, 研究方向为知识库与数据库、语义网、智能信息处理。

软件进化关心的是软件产品交付给客户之后对其的修改维护,是软件工程的一个子领域^[1]。对软件进化目前没有统一的定义,以下几种定义:

RISE (The Research Institute for Software Evolution)认为,一般的软件进化涉及到了软件交付给客户之后所发生的所有活动。它定义软件进化为一系列的活动,包括技术上和管理上的,这些活动能够确保软件产品不断达到商业要求的目标,且是低成本高效的完成。

Manny M. Lehman and Juan F. Ramil 定义软件进化为:有目的地从早期的可操作版本来产生新的软件版本的所有规划设计活动。

L. A. Belady 的定义为^[4]:软件系统在它们的生命周期里被维护和增强的动态行为。

Ned Chapin 定义软件进化为:它是软件维护活动和过程的一个运用,以及对这些活动和过程质量的保证和管理。这些活动和过程是用来从一个早期的可运行版本中来产生一个新的软件版本,这个新的软件必须满足客户要求改变的功能或属性。

尽管软件进化的定义没有统一,但其实质都是一样的,软件进化是一个过程,在这个过程中,程序要改变其形态来适应市场的要求和从先前程序中继承而来的特性。实现进化的最终目标是使软件能够更好地实现客户的需求,更好地适应环境的改变,使软件的功能不断地完善和增强。

2 软件进化的发展历程

在有计算机的早期,软件进化的行为主要就是给新的应用编写新的程序。直到20世纪60年代后期,人们才开始意识到旧的软件系统不能只是简单地被淘汰,软件需要被管理,软件的维护和进化是一个重要的活动^[1]。

现在人们普遍接受这个观点,反映现实世界应用的经常使用的软件必须持续不断地被改进和增强来维持人们对它的满意程度。这个观点第一次出现是被作为软件进化的原则陈述出来的^[5]。对这个观点,早在1968年在Garmisch会议上就被公开讨论过。同时,IBM公司于1968年开始了软件过程的研究^[2],Lehman等人在IBM中研究了规划设计过程,而用来检验和建模持续变化过程的一份研究报告被应用到了IBM的OS/360²操作系统上,这份报告是关于支持软件版本实现的计划和管理的工具等系统进化的很简单的模型。尽管这个模型的发展是相对简单的,但对它的研究促进了软件进化发展及软件进化原则的出现。1971年,人们第一次把对软件过程看作是一个反

馈系统(feedback system)进行了讨论^[6]。1979年开始,人们把软件进行了SPE三种类型的分类^[7],认为E类型系统解决的问题、从事的应用都是现实世界中的,是现实世界中模型的反映,并认为E类型软件必须要持续不断地进行进化。从1974年开始一直到1996年^[8],人们逐渐形成并完善了软件进化的8个基本原则,这些基本原则都是针对E类型软件的。

人们早期对于进化研究的数据是从IBM OS/360-370操作系统上获得的,随后是其他操作系统。早期的研究主要集中在进化的行为上^[2]。Lehman等人在对软件进化的研究中认为进化是大型程序的内在特性,每个特定系统都存在潜在的进化要求,这个现象可以被系统地研究和建模,且这些相关的模型可被用来预测未来系统的发展情况。

早期的研究在很大程度上不被计算机科学和软件工程组织重视,尽管如此,这个现象逐渐吸引了其他人的注意。由于程序的动态增长及其过程中各种因素的影响,1989年,出现了软件不确定原则^[6],随后是1993年FEAST(Feedback, Evolution And Software Technology)假设,该假设认为全球的E类型软件系统的进化过程是一个自我稳定的系统,一个复杂的多循环、多层次、多代理的反馈系统^[9],该假设是以前研究的一个总的反映。在1996年和2001年期间,人们开展了FEAST/1和FEAST/2工程^[6],这些工程是依赖于FEAST假设的,它研究了在E类型软件系统的进化和软件过程的改进中反馈的作用和影响。直到现在,软件进化一直被人们进行着广泛系统的研究。

3 主要的研究内容

现在,软件进化被人们广泛系统地研究着,人们对它研究主要是从两个方面来进行的^[10]。一个是把进化看作名词,主要是研究进化的本质、原因、特征、结果、影响、管理、控制等。如:什么是进化;为什么需要进化;什么驱动、控制和限制着进化;进化对成本、时间、进化过程的价值及产品的影响是什么等等的问题。

另一方面是作为动词来研究的,主要考虑的是进化过程的实现。包括提供和改进的方法、过程、行为、语言和工具等。比如:进化行为的目标是什么;为了达到目标,哪些地方需要进化;采用什么方法和技术比较合适;采用什么工具比较恰当;还需要哪些资源和技能等等。上述两方面是相辅相成的。加强对进化特性、管理和控制等的理解有利于过程管理和过程改进。对进化原因、属性和影响的研究有助于更系统有效地管理和制定计划,更有效地控制其过程和行为,这意味着更加需要有效的方法和合理的工具。而反过来,对于

软件进化方法、语言、工具、过程等的研究,又进一步帮助发现在软件进化理论和基础方面的缺陷和问题,有利于弥补进化理论基础方面的不足,有利于理论和框架的发展,并且有利于加速软件进化的过程。因此,软件进化的两个方面必须共同前进。

4 软件进化的方法

目前,快速增长变化的需求使得大多数的软件变得很难维护。然而,抛弃现在的系统而重新开始设计完成系统在经济和时间上是不可行的。因此,就迫切需要有效的方法来指导和控制它们的进化。但是,目前有效而系统的进化方法还不是很多,以下为几个典型的软件进化方法。

文献[11]提出了一种形式化的程序转换方法,该方法是基于一个形式化的语言 WSL(wide spectrum language)和方法的。这些转换在一个程序改变其形式的时候可保持其语义不变。它们被应用于重构系统以及抽象其中高层的表示。借助于一系列合适的转换,可以使这些抽象的表示等同于代码。

WSL 是这个方法的关键,因为所有的转换都用它来表示,它必须能够方便地表示底层的一些命令语句和高层的一些规范文档,且要能很容易地证明程序在一系列的转换中是否等价。利用该语言,可证明一个程序是否正确地实现了一个规范要求,一个规范是否正确地描述了一个程序的行为。该方法首先将给定的一个原程序 P 转换为一个用 WSL 语言表示的等价形式 P'。然后从 P' 开始,从一个预先被证明好的转换库中选择一个转换,这将可能产生新的中间形式 S_1 , 然后,接下来的一系列转换将使软件被转换成 S_2, S_3, \dots, S_n 。最终的结果是要依赖于使用者需要的,比如希望只是简单的重构还是需要提炼出一个抽象的规范说明。

这个方法中有三种基本的操作:程序转换,细化和抽象。转换操作可以修改一个程序变成一个不同的形式,但却有着同样的外部行为;细化使一个程序的行为更加被确定化;而抽象操作与细化相反。其更详细的描述可参考文献[11]。

Maritta Heisel 等人也提出了一种指导软件进化的方法^[12],这个方法依赖于一系列在需求和代码之间由映射连接的中间制品(artifact),这些中间制品及其连接可以被构造和维护,它们可以用一个系统的方法来指导软件的进化过程。

该方法把软件开发中的问题和解答领域分离开来,问题领域包含了软件的需求,对软件的内部没有涉及,解答领域包含了程序代码以及实现代码时所建立

的文档。

问题领域概念是指在问题领域中有意义的一些概念(如软件将要模拟和交流的环境等)。这些概念可以在需求中出现,如一些对象等。但还包括一些需求中没有的领域知识,它们是需求中有关概念的隐性信息,但在规范说明的编写中非常有用。而解答领域概念是在抽象代码和软件架构中使用的(如图 1 所示)。

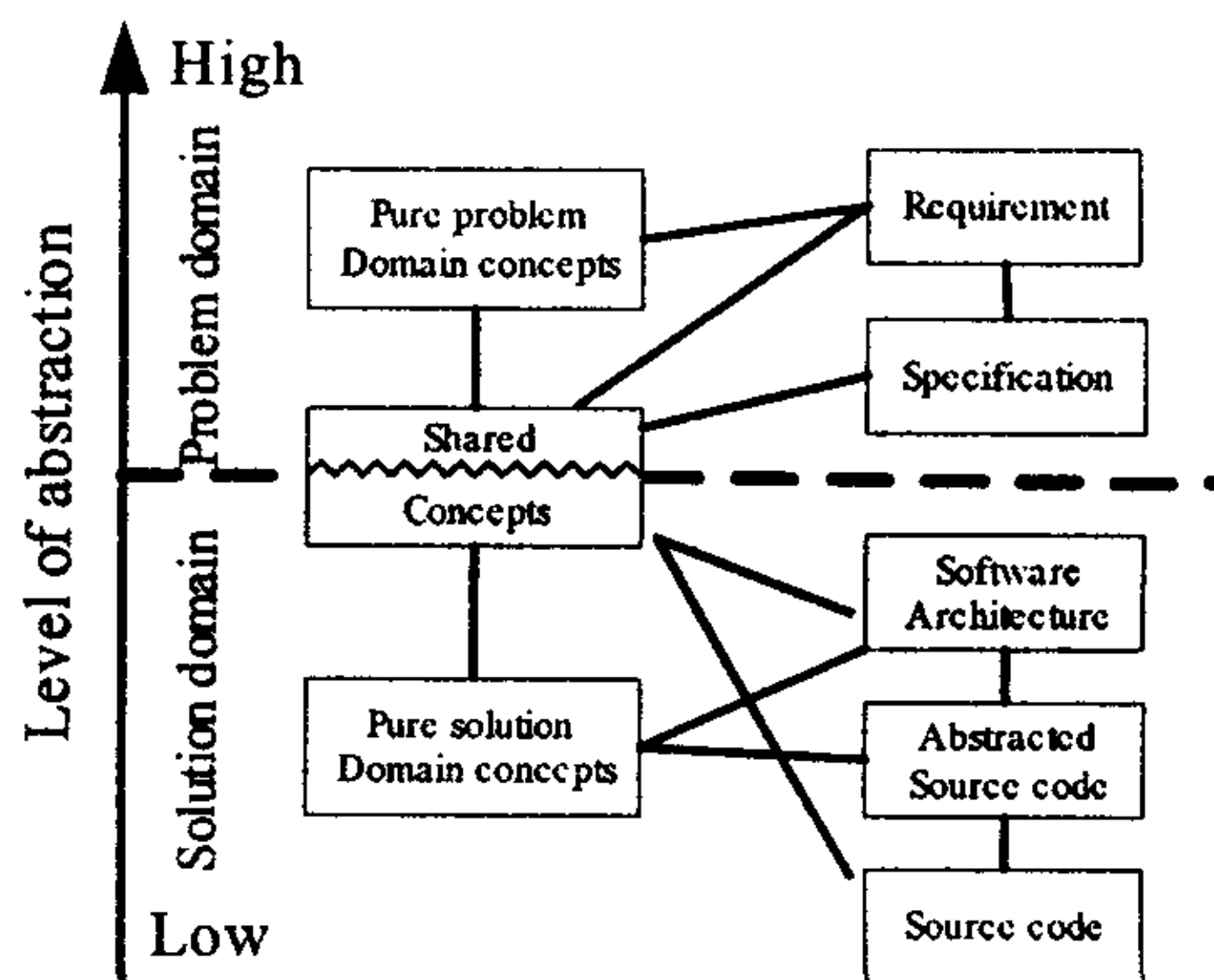


图 1 Artifacts and mappings

规范说明是一个可实现的需求,但要避免过多的实现细节,它描述了软件对外的接口,不考虑其内部工作。它只允许使用在问题领域中有表示的概念,因此,和纯粹的问题领域概念无直接关联。而抽象代码对于理解源代码的原理很有帮助,且易于文档化。但达到一定的高度时,抽象代码就不再合适了,这就需要软件构架来更好地描述其工作原理,软件构架主要考虑了软件的内部结构和工作原理。同时,两个领域间必须要有一个联系,至少两者概念可以互相转换,而共享的概念则实现了将两者相匹配的概念合并的功能。但这个匹配过程的关键是不同领域中概念的语义必须相匹配。而那些没有匹配的概念则被划到各自的纯粹领域中。

文献中描述了如何构建这些映射及一些相关确认规则。一旦软件要改变的要求被确定下来,首先会影响到问题领域的概念,使原有的概念作废或添加新的进去,而这些受影响的概念可被概念和需求间的映射所捕获。同样,需求的改变多少都会影响到共享概念,有些是直接的,有些是通过纯粹问题领域概念来间接影响的。这样逐层往下,所有的改变都可影响到代码层。为了支持这种自顶向下的结构,标签被引入进来。一种改变标签和每个要改变的需求、概念和结构元素相关联,每个改变标签包含了要必须修改的一些描述。另一种违例标签被自动创建来检查是否在这个过程中违反一些确认规则。在这个过程中,标签可用来决定是否一个底层的中间制品要考虑被修改。根据改变标签上的线索可以知道改变的过程,确认标签同样帮助

了这个过程。这样,就使整个软件进化过程从上到下可以有条不紊地进行,这种方法有效地指导了软件进化的过程。

唐亚哲等人提出了一种基于开放实现与反射的软件进化模型^[13]。传统的软件扩充机制,如基于源代码的软件扩充和二次开发扩充等方法,要么信息量太大,要么适用条件有限,都不能得到广泛的应用。开放实现与反射技术从一种全新的角度和系统化、结构化的方法为软件进化提供了新的方法和手段。与以往方法相比,基于开放实现与反射技术的软件进化不仅为软件进化和扩充提供了清晰的接口,且由于较好地实现了分离原则,大大降低了软件扩充的风险和提高了软件质量。

开放实现是与传统软件开发的“黑盒”原则相对的,简单说,开放实现就是模块或对象不仅提供功能接口,而且提供定制接口(就是配置接口,对模块或对象内部具体实现的某些方面进行设置)。定制接口的提供有各种方式,其中最主要的就是元对象协议 MOP (Meta Object Protocol)。MOP 是反射技术术语,与元对象相对的是基对象。它是正常系统应用对象,而元对象是表达基对象如何实现的对象,即元对象控制基对象的操作。通过该技术可实现软件运行时动态变化。就是说系统不仅向用户提供功能,且提供用户进行高级定制的接口,并在系统内部将这些定制接口与系统功能进行因果关联。

该方法的研究基于这样一个认识:系统的扩充和进化不是随意的,而是可以在相当程度上预见的。其目的就是尽可能多地预见系统的进化,并用一种有组织、结构化方法来实施这种进化。同时,方法引入了“软件运行进化”的概念,并将其形式化定义为六元组 C。

$C = \{\text{源代码, 二进制代码, 编译器, 解释器, 用户配置, 二次开发}\}$

显然,六元组与软件生命周期的各个阶段有关。在设计阶段就要明确系统的进化特点,比如哪些方面容易变化等,最好能详细记录这些信息。在源代码设计阶段,可以使用反射语言或定义元对象来支持进化。

这种方法有利于帮助进化。首先,它们强调分离原则,将大量非功能与功能属性用不同的模块来表达,它们的耦合关系由自动工具来实现,减少了程序员进行程序阅读和修改的工作量,并提供可靠性。其次,该技术将原本由程序逻辑隐含表达的系统内部实现细节用元层对象表达,提供了基层程序员操纵程序实现细节的高层接口。第三,该技术提供了一种结构化的方法来实现软件动态扩充,使软件扩充的整个过程处于

严格控制之下,软件的修改和扩充不再是一种随意性的活动,确保了软件进化的质量。

以上提到的三种支持进化的方法中,第一种只是对程序进行了等价转换,其转换操作可以修改程序变成不同的形式,但却有着同样的外部行为。因此,这种方法对于添加新的功能和删除一些功能有些限制,这种方法主要是集中在了代码层的进化上,有一定的局限性。第二种方法从宏观的方面对软件过程进行了跟踪和进化,从软件工程的角度对进化进行了控制,从最初的需求改变直到最后的编码设计,都有效同步地控制了进化的过程。最后一种方法,可以说是对软件的进化进行了预处理,提前就分析了系统的进化特性,在设计实现的时候提供了充分的用户定制接口,大大降低了软件进化的难度。

5 面临的挑战和问题

尽管人们对软件进化及其过程进行了多年的研究,也提出了一些相关的理论、方法和工具来支持软件的进化。但是在软件进化的过程中,仍然面临一些挑战和问题^[14]。

(1)软件进化技术应该被提高到一个更高的抽象层面上。目前,现存的支持进化的工具大都定位在程序层面,对高层的支持很少。而反过来那些支持高层设计和建模活动的工具对软件进化提供的支持又是很少的(如 UML CASE 工具)。这就存在一个问题,在设计模型(包括文档,构架和规范说明等)和原代码之间很难同步进化。一般应该使一个层面上的改变能反映到其他层面,保持进化的同步性。

(2)软件开发过程应该支持软件进化。人们需要研究如何将软件改变的思想结合到传统的软件开发过程中。一个典型的方法是采用迭代的和增量的软件开发过程。尽管有一些过程模型被认为对软件进化提供了显性的支持(如 staged life-cycle model),但对其研究还不是很深入,人们还是要研究哪种软件过程更适合哪种类型的软件系统。

除了要对进化过程模型更好地理解和支持外,还需要提高进化过程中管理的意识,像软件工程一样重视管理,加强对计划、组织和控制软件进化过程的管理。

(3)软件进化中需要更多的经验研究。在软件进化领域中,存在对于更多经验研究的需要,有时为了获得重要的统计结果,就需要一个足够大的有代表性的案例集,这在工程领域中并不总是容易得到的,软件进化为了获得这些数据是十分困难的。一个可行的办法就是根据以往经验,对于要解决的这类问题进行案例

学习,根据以往的经验提出一个进化的基准。

除了以上提到的外,在软件进化中还存在一些其他问题,如随着软件的进化,如果没有主动的策略,软件系统的质量会逐渐下降,这在很大程度上是由外部的原因导致的(比如经济压力等)。再如一些技术方面的问题等。因此,对于存在的这些问题,需要基本的关于形式和理论的科学研究来帮助理解、分析和管理软件的变化。需要语言、工具、方法和技术的发展来提供对于在软件开发过程中软件改变的显性的支持,这是需要人们做大量研究工作的。

6 结 论

尽管人们对于软件工程的研究使得软件的开发从各个方面有了提高,但是,软件的质量仍然达不到人们的期望,软件要不断面临新的需求和环境而改变。因此,软件进化一直是人们研究的一个重要领域。笔者简单介绍了软件进化的几个方面,由于进化及其过程涉及到了很多的方面,因此,对其的研究必须覆盖各个方面,如理论模型的研究、过程模型的进化、技术工具方面的研究等。同时,人们应该使其在传统软件开发过程中得到重视,对其提供各种显示的和直接的支持研究。

参考文献:

- [1] Bennett K. Software evolution: past, present and future[J]. Information and software technology, 1996,38:673-680.
- [2] Lehman M M, Ramil J F. Software evolution - Background, theory, practice [J]. Information Processing Letters, 2003, 88(1/2):33-44.

(上接第 195 页)

实例化(预计算并存储结果)Cube 中的部分节点以提高 OLAP 性能的解决方案。

利用决策树分类方法^[4]和 Bayes 分类器^[5]等数据挖掘方法对数据仓库进行挖掘分析,得出有用的规则,用来改进企业的软件过程、对正在实施的项目进行更合理的评估和支持企业的决策分析。

3 结束语

问题跟踪工具为软件企业的软件开发过程中问题的解决提供一个良好的平台,问题分析系统为企业提供了有用的决策信息。问题跟踪和问题分析系统对提高软件生产率和软件质量起到了一定积极有效的作用。相信随着该系统的不断完善,高质量的分析型数据的日积月累,问题分析系统将会发挥越来越大的作

- [3] Lehman M M, Ramil J F. Software evolution and software evolution process[J]. Annals of software Engineering, 2002, 14:275-309.
- [4] Belady L A, Lehman M M. A model of large program development[J]. IBM Systems Journal, 1976,15(1):225-252.
- [5] Lehman M M. Programs, Cities, Students, Limits to Growth? [J]. Imperial College of Science and Technology Inaugural Lecture Series, 1974,9:211-229.
- [6] Lehman M M, Ramil J F. Tutorial on Software Evolution: its Source, Nature and Control[C] // ICSM 2002. Canada: [s. n.],2002.
- [7] Lehman M M. Programs, lifecycles and the Laws of Software Evolution[J]. Proc. IEEE,1980,68(9):1060-1076.
- [8] Lehman M M. Laws of Software Evolution Revisited[C] // EWSPT. France:[s. n.],1996.
- [9] Lehman M M. Feedback in the Software Evolution Process [C] // CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics. Dublin:[s. n.],1994.
- [10] Lehman M M, Ramil J F, Kahen G. Evolution as a Noun and Evolution as a Verb[C] // SOCE 2000 Workshop on Software and Organization Co - evolution, Imp. Col., London:[s. n.],2000.
- [11] Ward M, Bennett KH. Formal Methods to Aid the Evolution of Software[J]. International journal of software engineering and knowledge engineering, 1995, 5(1):25-47.
- [12] Heisel M, von Schwichow C. A method for guiding software evolution[C] // The IASTED International Conference on Software Engineering. Innsbruck, Austria:[s. n.], 2004.
- [13] 唐亚哲,陈传峰,李增智. 基于开放实现与反射的软件进化模型[J]. 小型微型计算机系统,2003,24(11):1978-1981.
- [14] Mens T. Challenges in Software Evolution[C] // ECRIM - ESF workshop ChaSE. Bern, Switzerland:[s. n.], 2005.

用。

参考文献:

- [1] Lonchamp J. A Structured Conceptual and Terminological Framework for Software Process Engineering[C] // In 2nd International Conference on Software Process (ICSP2). Berlin, Germany:[s. n.],1993:41-53.
- [2] Grimshaw A S. Easy - to - Use object - oriented parallel processing with Mentat[J]. IEEE Computer, 1993,26(5):39-51.
- [3] 于波,赵征,唐世渭. OLAP 中的 CUBE 计算问题[J]. 计算机应用,2003,23(1):1-3.
- [4] Han Jiawei, Kamber M. 数据挖掘:概念与技术[M]. 北京:机械工业出版社, 2001:188-195.
- [5] 边肇琪,张学工. 模式识别[M]. 北京:清华大学出版社, 2002:9-43.