ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.3, March 2006, pp.620−627
DOI: 10.1360/jos170620

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

面向 Aspect 的操作系统研究*

陈向群+, 杨芙清

(北京大学 信息科学技术学院,北京 100871)

Research on Aspect Oriented Operating Systems

CHEN Xiang-Qun+, YANG Fu-Qing

(School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: +86-10-62757160, Fax: +86-10-62753996, E-mail: cherry@cs.pku.edu.cn, http://www.pku.edu.cn

Chen XQ, Yang FQ. Research on Aspect oriented operating systems. *Journal of Software*, 2006,17(3):620−627.
http://www.jos.org.cn/1000-9825/17/620.htm

Abstract: Aspect-Oriented software development (AOSD) is a new type of software design idea and technique. This paper analyzes recent research results in three fields which are: (1) the crosscutting concerns and Aspect concept in operating systems; (2) component reconstruction, system evolution and design, and system security; (3) performance monitoring and fault tolerance. The paper points out that some positive research results of Aspect-Oriented operating systems have been acquired. However, the current research results lack necessary deepness and extent, the operating systems that adopt the Aspect idea during design stage do not exist yet, and there is no whole engineering and standardization for Aspect extraction procedure in the existing operating systems code. The solutions for these problems rely on further Aspect-Oriented research results. Final, the paper describes the respective of Aspect oriented operating systems research and indicates that the research of AOSD may possibly bring a significant impact to the research and development of future operating systems.

Key words: Aspect-Oriented software development (AOSD); Aspect-Oriented programming (AOP); operating system

摘 要: 面向 Aspect 软件设计是一种新的软件设计思想和技术.分析了近年来操作系统贯穿特性与 Aspect 概念,构件重构、系统演化与设计,系统安全、性能检测与容错这 3 个方面的研究成果,指出面向 Aspect 操作系统研究已经获得了积极的成果.但是,目前的研究缺乏一定的深度和广度,尚没有在操作系统的设计阶段运用 AOP(Aspect-Oriented operating)思想的成果出现.在已有操作系统代码中抽象 Aspect 的过程中,缺乏完整的工程化和规范化的研究.这些问题的解决有赖于面向 Aspect 研究的进一步深入.最后,对面向 Aspect 操作系统研究的前景进行展望,认为有关 AOSD(Aspect-Oriented software development)的研究有可能对未来操作系统的发展产生重大影响.

关键词: 面向 Aspect 软件设计;面向 Aspect 程序设计;操作系统
中图法分类号: TP301 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant No.60373001 (国家自然科学基金)
Received 2004-05-18; Accepted 2005-08-24

在计算机软件设计过程中,一个需要遵循的重要原则是清晰地分离各种关注点(separation of concerns),把涉及多方面的复杂问题加以分解,化繁为简、化大为小,然后逐个处理,最后形成一个完整的对原有复杂问题的解决方案.但是,由于在软件和操作系统的代码中存在多重贯穿特性(crosscutting concerns),产生代码交织(code tangling)和代码散布(code scattering)的现象,影响了模块的内聚性和模块之间的独立性.对于代码中多重贯穿特性持续深入的研究,导致在 20 世纪 90 年代出现了面向 Aspect(Aspect-oriented programming,简称 AOP)程序设计技术[1].

为了实现面向 Aspect 程序设计,需要在面向对象程序设计语言的基础上,提供对 Aspect 的支持,这就是面向 Aspect 程序设计语言.第一个通用 AOP 程序设计语言 AspectJ[2],由 Xerox PARC Kiczales G 领导的研究小组经过 8 年的研究于 1997 年实现.它构建在 Java 语言之上,是 Java 语言的扩展[1].目前,国际上已经发布了多种 AOP 程序设计语言,如 AspectC[3],AspectC++[4,5],IBM HyperJ[6]以及 AspectR[7],AspectC#[8]等.面向 Aspect 软件设计(Aspect-oriented software development,简称 AOSD)的出现不过短短几年的时间,都已成为软件领域中的一个重要研究热点,有关面向 Aspect 操作系统的研究工作也在逐渐展开.

本文第 1 节以操作系统中的贯穿特性与 Aspect 概念,构件重构、系统演化与设计,系统安全、性能检测与容错这 3 个不同主题,对近年来国际上面向 Aspect 操作系统研究的主要成果给予介绍和分析.第 2 节对有关研究成果加以总结,分析存在的问题,并提出对未来面向 Aspect 操作系统研究前景的展望.

1 面向 Aspect 操作系统研究

在操作系统中有大量贯穿特性存在,其中有高层次的概念,如安全、服务质量等,也有较低层次的概念,如缓冲、登录、同步等.它们既无法用传统的面向过程程序设计加以抽象,也不能用面向对象技术进行准确和高效的抽象,只能通过设计人员的跨结构(包括跨子系统、跨层次、跨模块或跨构件)设计,即对某一贯穿特性所涉及的子系统、层次、模块或者构件逐个进行具体的程序编写和调试,才能实现该贯穿特性所预定的设计目标.

已经发表的研究成果说明,将 AOSD 运用到操作系统领域中,能够提高操作系统中贯穿特性的模块化程度,有利于改善操作系统的可配置性以及移植性,提高设计和开发效率.

1.1 贯穿特性与Aspect概念

Coady Y 和 Kiczales G 等人研究了 FreeBSD v3.3 操作系统中的贯穿特性以及 AOP 在客户机/服务器体系结构中的作用,发表了若干篇论文[9−13],这是较早报告面向 Aspect 操作系统研究的一批论文.

页面预取是操作系统内核中贯穿特性的一个典型.在 FreeBSD 中,页面预取包含 4 个环节:预测、成本分析、计划和实际预取页面操作.在虚拟存储器系统(VM)和文件系统(FFS)中,分别用各自的判据对所需页面进行预测;成本分析环节涉及了一些低层的因素,诸如磁盘访问的代价、盘上数据的连续性、数据的位置等,并提出在最佳成本-效益情况条件下应该预取的磁盘块;而计划环节则把预测和成本分析组合起来,以确定应该从磁盘上预取哪块数据;实际预取环节则执行计划,向对应磁盘发出页面访问操作命令.页面预取的过程构成了一条专门的执行路径,它不属于 FreeBSD 操作系统中的任何一个层次,而是贯穿了系统中各个有关层次.页面预取方式的这种贯穿特性使得预取方式难以采用传统技术进行模块化.在 FreeBSD 的实现过程中,页面预取代码有 265 行,组成了 10 个连续代码块,分散在系统 3 个层次的 5 个函数中.可见,其模块化效果是很差的[13].

FreeBSD 中的原有代码和用 AOP 实现代码的关键差别在于:在运用 Aspect 之后,虚拟存储器 VM 和文件系统 FFS 的预取行为变得清晰了.低层次的预取代码可以准确地为高层次预取代码服务.进而,在页面预取的 AspectC 实现中,每个页面预取代码都被局部化在单一的 Aspect 内.这样,就可以实现预取模式的即插即用,即可以将每个 Aspect 安排在单一的文件中,并运用标准的 makefile 技术添加或排除某个特定的预取模式[13].

文献[10]研究了客户机/服务器系统中的缓冲一致性和页面预取问题,并得出如下结论:在引入 AOP 技术之后,AOP 使客户机/服务器系统更易扩充,核心代码更稳定,且可以使所扩充的功能实现模块化.

PURE 小组[14]采用 AspectC++语言重写了面向对象操作系统 PURE 的中断同步代码.对 PURE 有关源代码

的分析表明,涉及中断同步的方法调用有 166 个,分布在 15 个类中,该中断同步是典型的贯穿特性.在实现了该中断同步策略的 AOP 设计后,源代码更加紧凑了,系统向真正的构件化操作系统方向前进了一步.该小组建议应该在操作系统设计的同时,设计构件代码和 Aspect 代码,这样可提高设计效率,简化未来的系统维护问题[14].

文献[15]提出了基于事件的 AOP(event-based AOP).这是一种 AOP 的通用操作模型,它利用事件提供了一种自然的 AOP 抽象,复杂的贯穿特性可以用事件的各种模式以及相关的种种通用操作来加以定义.文献[15]认为,Aspect 可以作为完全基于 AOP 操作系统内核架构的建造块,从而提高操作系统的配置性和可靠性.

分析对已有操作系统贯穿特性研究方面的论文,可以从中总结出一个共同的思路:首先,确定一种将要进行研究的操作系统,选择的前提是研究人员应该已经对该系统的源代码进行过深入的研究,或者该系统本身就是由研究小组的成员设计的,研究人员对其结构和源代码非常了解;其次,在该系统中选择某个无法用常规程序设计语言或面向对象语言实现模块化的功能或特性,即所谓贯穿特性,对所确定的贯穿特性的代码进行深入的分析,研究其代码散布或缠绕的程度;第三,研究有关的贯穿特性是否确实具有 Aspect 的基本特征,能否使用适当的 AOP 工具,对该贯穿特性的有关代码按照面向 Aspect 思想进行重新设计或重构;最后,对重构前和重构后的系统进行分析对比,可以比较代码规模和运行效率等方面的变化,研究重构后贯穿特性代码的模块化程度以及其他感兴趣的问题,最终得出结论.

在操作系统领域中,研究人员使用的 AOP 工具主要是 AspectJ,AspectC 和 AspectC++等 AOP 语言.一些研究人员从操作系统研究需要的角度对面向 Aspect 概念、语言和应用过程进行了更深入的研究.

PURE 研究小组发现,AOP 技术可以改进 C++内联(inline)功能的使用效果[16];文献[17]对改进 Aspect 与构件之间的接口提出了建议;VEST 研究小组提出了独立于语言工具的 prescriptive Aspect 概念,并将 AOP 思想运用到一个实时嵌入式操作系统的设计阶段中,取得了较好的效果[18];文献[19]针对操作系统的特点,研究设计了一种用于 C 语言系统编程的 Aspect 语言 Arachne,用以更好地描述操作系统中的贯穿特性,并验证了新的 Aspect 语言 Arachne 可以在需要高性能动态响应的 Squid web cache 中使用.

需要指出的是,有关 Aspect 概念与 AOP 语言的研究是 AOSD 研究领域的一个极为重要的方面,已经有大量的研究论文发表,这里介绍的只是一些直接涉及操作系统领域的研究成果.

1.2 构件重构、系统演化与设计

文献[16]还探讨了用 AOP 技术改造面向对象 PURE 操作系统内存管理构件的效果,即在最好情形下,内存管理构件代码减少到原有的 33%,执行时间减少到原有的 30%;在最差情形下,代码规模也至少有 20%的降低.而在一些特定情形下,高可配置的、面向对象 PURE 操作系统中资源的开销甚至小于在同样硬件环境上用手编汇编代码实现的系统[20].

从 AOP 机制可以实现贯穿特性的模块化这一点出发,Coady Y 和 Kiczales G 对 FreeBSD 操作系统 3 个版本中的 4 个贯穿特性进行了基于 Aspect 的改写.将有关 4 个贯穿特性的原有代码版本与运用 Aspect 技术实现之后的版本进行比较.研究人员发现,面向 Aspect 技术对系统内核的演化性、可配置性、代码的冗余性以及模块的可扩展性等这 4 个关键性质都有改善的作用[21].

Bossa 小组[22]研究了 AOP 对采用 Bossa 调度程序架构的 Linux 内核演化的作用.按照常规的手工方法,要在 Linux 内核中实现一种 Bossa 调度程序架构,需要生成 300 多个事件,要对 27%的系统服务以及已有驱动程序中的 15%进行调整,这是一项令人厌烦和容易出错的工作.Bossa 小组将基于事件的 AOP 与时序逻辑相结合,用一组变换规则来表达 Aspect,完成了 Linux2.4 内核的再工程,实现了 Linux 内核自动调整与演化.

AOSD 会对操作系统设计和体系结构带来什么影响?文献[23,24]提出,操作系统应该是一个面向 Aspect 的三维构架,包括构件维、Aspects 维和层次维.同时,应该把 Aspect 与其策略分离,从而支持高度的复用性.

文献[25]指出,软件中的一些非功能属性是不能用软件的基本功能来描述的,但它们又是能被最终用户观察到的软件运行行为.性能和资源的利用效率是软件非功能属性的典型例子,而操作系统的体系结构自身也是一种非功能属性.不过,软件体系结构与许多非功能属性有关,它是一种全局 Aspects(holistic Aspects),即它的非功能属性的实现并不仅仅体现在代码的贯穿特性上,而且与软件开发的整个过程有关.

由于修改系统体系结构的代价极大,文献[25]指出,应该运用 AOSD 思想,从操作系统设计的开始阶段就把有关的贯穿代码通过 Aspect 来处理.该文简要叙述了正在进行的 CiAO 研究项目,其目标是运用 AOSD 思想与技术,实现具有可配置体系结构属性的一个操作系统系列.

1.3 系统安全、性能监测与容错

安全性能被人们看作是系统软件中典型的贯穿特性.在有关面向 Aspect 研究工作的初始阶段,一些论文往往只是在概念上叙述这一点,所举出的有关面向 Aspect 的安全例子也是非常简单的,不具备实际的应用意义[26].随着研究工作的逐步深入,这种状况已经发生了改变.有必要指出,本节所论述并分析的有关系统安全研究工作并不都与操作系统直接相关,但是这些研究成果对于操作系统的安全性能设计是有参考意义的.

文献[26]论述了在一个 FTP 服务器协议上进行的安全性能研究工作.该文采用 AspectJ 语言完成了安全性能的 Aspect 实现.文献[27,28]具体分析了对访问控制的面向 Aspect 设计以及面向 Aspect 访问控制的通用化设计,建议每个 Aspect 对应安全上的不同需求,而每个 Aspect 的实现都取决于 Aspect 下面的不同安全机制.这两个文献提出将所有的安全 Aspect 组合起来,构成一个系统安全 Aspect 基础.文献[29]将面向 Aspect 的 C 语言运用到系统安全性能的设计上,取得了良好的效果.该文认为,面向 Aspect 技术使得安全策略能够与代码分离开来.这样,开发人员可以只专注于编写程序代码,由安全专家定义恰当的安全策略,从而有助于安全性能水平的提高.

文献[26]还叙述了研究过程中所遇到的困难.这些困难源于原始代码并没有考虑到要分离安全逻辑,从而导致了分离安全逻辑上的困难,其直接结果是有关安全性能的 Aspect 设计实现并不优雅.原有代码规模为 154.3 KB,而把有关安全逻辑迁移到 Aspect 中之后,代码规模增加为 163.4KB,然而这些困难是可以克服的.

在系统安全领域,一个值得重视的方面是有关 AOSD 对提高系统抗攻击能力的研究.文献[30]研究了 AOSD 在网络通信过程中防止恶意篡改代码的作用.该研究工作既使用了静态 Aspect 技术,也使用了动态 Aspect 编排技术.这样,可以随着抵抗攻击的需要,插入或者撤除不同效能的 Aspect,从而完成必要的安全检查.文献[31]运用 Aspects 技术,使一台遗产客户机/服务器软件提高了抗“消息篡改”攻击的能力.该文认为,Aspect 技术可以改变系统的安全策略,实现系统安全边界可变性,从而有利于未来系统安全的演化.

文献[32]叙述了一个面向 Aspect 的安全构架(Aspect-Oriented security framework,简称 AOSF)设计.该文献认为,AOSF 具有分离关注点、前瞻性、整体性、一致性、可适应性以及无缝集成等基本特性,并叙述了在实现该 AOSF 工作中的经验教训.

可以认为,AOSD 能够实现系统安全性能的模块化,使安全代码与安全策略分离,从而有利于系统整体安全性能的设计和改进.将具有不同安全策略的 Aspect 组成一个安全构架的思想,有助于提高系统安全性能的整体水平.

AOSD 既然可以对贯穿系统内部各个模块、散布在系统各处的代码进行模块化处理,那么,有意识地在系统内部的构件或模块中插入与性能监测相关的代码,对系统整体性能进行实时监测也是可行的.

文献[33]指出,使用 Aspect 技术实现程序的自动监测有 3 种方式:首先,可以设计出可控制的出错调试工具,实现只有在源代码出错时才触及或调整源代码,这就避免了一些偶发性的错误;第 2 种方式是进行程序测度,得到程序中的函数或对资源的利用随时间变化的数据.这些数据对优化程序、提高系统整体性能非常有价值;Aspect 程序监测的最后一种方式是对程序运行进行监视和检查.这里,监视是指在外界可以看到所收集的程序运行内部数据,而检查则意味着要将收集的数据与预定的数值加以比较,在达到某个值时才以某种方式反应.

文献[33]报告了 3 个具体的程序监测研究实例:第 1 个是在 Pentium 系统中进行任务监视;第 2 个是监视存储管理系统,用于查找内存泄漏,了解应用如何使用内存,从而可以协助优化内存的管理;第 3 个是对 PURE 构件化操作系统进行运行检查,以便确认实时环境中的构件是否在规定的时间内完成了预定的功能.

文献[34]运用 AspectJ,在已有系统中对异常处理的代码进行简化,处理一些经常发生的异常事件.研究人员发现,使用 AspectJ 减少了原有异常检测和处理的代码数量,在最好的情形下,新代码数量只有原有代码的 1/4,其原因是减去了原有 Java 代码中的冗余.该文献认为,AspectJ 提供了对异常行为处理的不同结构、更好的容错

方式、更好的代码重用以及更清晰的程序文本.

与一些有关容错研究主要采用 AOP 静态特性处理故障代码不同,考虑到在系统运行过程出现的故障具有动态特性,文献[35]描述了运用动态 Aspects 技术设计的一个系统,称为 Rescue.该系统可以确定或发现程序运行中的各种故障,然后采取措施进行故障恢复.

运用面向 Aspect 思想和技术,对有关系统性能监测和容错处理的程序代码进行 Aspect 模块化设计的思路是值得重视和深入研究的.这种方法不但有可能提高设计和实现操作系统性能监测、异常处理和容错处理的效率,而且有可能开发出新的操作系统实时性能监测工具,动态地确定或发现程序运行中的故障,并实现故障恢复,从而提高系统整体的安全性和健壮性.

2 结论、问题与展望

2.1 结 论

综上所述,面向 Aspect 操作系统的研究工作已经取得了积极的成果:首先,对已存在的操作系统代码进行分析,提炼其中的贯穿特性,然后运用适当的面向 Aspect 设计工具,对贯穿特性进行模块化,或者对有关的操作系统构件进行重构,能够提高操作系统的模块化程度和设计效率;其次,在操作系统演化、体系结构设计以及系统安全等方面,AOSD 为这些方面的研究和设计工作提供了新的解决思路.

进一步分析可以看到,在面向 Aspect 操作系统的研究中有两种不同的技术路线:第 1 种是对已经存在的操作系统代码进行面向 Aspect 分析,提炼出其中的贯穿特性,然后运用适当的面向 Aspect 设计工具对有关的操作系统代码进行重写,或者对有关的系统构件进行重构,从而达到预期的目的.目前有关面向 Aspect 操作系统的研究多数采用的是第 1 种路线;第 2 种则是从操作系统的设计阶段开始,运用面向 Aspect 的思想和技术,对操作系统进行基于向 Aspect 的全新设计.显然,第 2 条路线困难更大,目前尚未有成功的先例.

2.2 问 题

尽管面向 Aspect 操作系统的研究取得了一些成果,但还是有大量的问题没有得到解决,有些潜在问题可能尚未意识到:

首先,在面向 Aspect 操作系统方面,有一定研究深度和广度的论文并不多见.从研究的深度来看,多数论文只针对某个操作系统中的一个或若干贯穿特性进行研究,对 AOSD 作用的分析还停留在定性层面上,缺乏严格的定量分析;在研究的广度方面,一些 AOSD 研究工作围绕着研究者自行研制、熟悉的操作系统展开,其他研究人员很难深入了解或熟悉所论及的操作系统并展开相关的验证或进一步工作,所以这类研究成果缺乏较强的说服力.不过,目前已有少量研究工作开始涉及主流操作系统,如 Linux,但是还没有针对 Unix 以及 Windows 操作系统的研究成果出现.

其次,现有的研究工作主要围绕已有的操作系统展开,还没有看到从操作系统的设计阶段就运用 AOSD 思想进行 Aspect 设计实现的成果报告.值得关注的是,CiAO 项目[25]已经以此作为目标开展研究工作.

第三,在从已有操作系统代码中抽象 Aspect 的过程中,缺乏完整的工程化和规范化的研究,对贯穿特性的识别工作依赖于研究人员对某个操作系统代码的熟悉和了解程度.大量存在的手工识别和分析 Aspect 的工作不利于 AOSD 思想在操作系统领域中的研究工作的开展.对于“面向 Aspect 思想和技术究竟对操作系统有什么作用”这一基本问题,现有的回答是不充分的,缺乏全面、系统、有深度的研究.

上述种种问题之所以存在,一个主要原因是由于操作系统的设计与实现本身就比较困难,而要对操作系统中的贯穿特性进行分析也是相当困难的.双重的困难给面向 Aspect 操作系统的研究带来了挑战.在操作系统领域中,研究人员在采用新设计思想方面更为谨慎,因为操作系统中一点点小的差错就有可能带来较大的危害.

另一主要原因是,面向 Aspect 思想和 AOP 语言本身还没有完全成熟.从 1997 年面向 Aspect 的经典论文[1]发表到现在,该论文中列出的许多需要研究的问题都还没有答案,有关概念的严格定义或形式化描述的研究工作也都还处在初步阶段.

2.3 展 望

目前,国际上有关 AOSD 领域的研究工作极为活跃[36].整个 AOSD 的研究工作已经从纯学术的理论研究转向学术研究和应用研究并行发展的态势.

在 AOSD 领域,最重要的国际会议是一年一度的 AOSD 会议(International Conference on Aspect-Oriented Software Development)[37].其中,“ACP4IS: Aspects, Components, and Patterns for Infrastructure Software”工作会议涉及在应用服务器、中间件、虚拟机、编译器、操作系统以及其他软件中的 AOP、构件模型以及设计模式研究;“LATE: Linking Aspect Technology and Evolution”工作会议则重点探讨“如何从已有软件中识别贯穿特性,如何分离它们”等.而“Early Aspects”工作会议的重点则是研究在软件生命周期的早期阶段,包括需求分析、领域分析以及体系结构设计等阶段中 Aspect 的作用.

值得指出的是,与前述第 1 种技术路线有关,针对如何在已有软件代码中识别 Aspect 的问题,已经形成了一个 AOSD 的分支研究领域:Aspect 挖掘与逆向工程(Aspect mining and reverse engineering)[38].出现了一些协助在遗产软件中进行 Aspect 分析与识别的工具,如 Aspect Browser[39],它可以在 Eclipse 环境中对软件进行贯穿特性的识别.Aspect 挖掘与逆向工程研究的进展,将推动相关研究从手工分析方式逐步转为以自动分析工具为主的方式.对动态 Aspect 编排技术的研究,将推动操作系统动态可配置水平的提高[40].

有关早期 Aspects(early Aspects)、面向 Aspect 建模以及软件体系结构中的 Aspect 等研究工作,势必会有利于面向 Aspects 操作系统研究第 2 种技术路线的实现.如果能够在操作系统的设计阶段考虑到操作系统中的各种贯穿特性,同时设计实现构件代码和 Aspect 代码,那么将有可能提高操作系统整体性能,便于操作系统的维护和演化,甚至于有可能引发新一代操作系统的出现.

回顾软件技术的发展历史可以看到,软件技术思想与操作系统密切相关,软件技术的发展推动着操作系统不断地向前发展[41].显然,Aspect 思想与技术的发展,为操作系统的研究和设计开发工作提供了新的思路和研究空间.整个 AOSD 的发展必然会对操作系统的未来发展产生重大的影响.

References:

[1] Kiczales G, Lamping J, Mendhekar A. Aspect-Oriented programming. In: Aksit M, Matsuoka S, eds. Proc. of the European Conf. on Object Oriented Programming. Berlin: Springer-Verlag, 1997. 220–242.

[2] Kiczales G, Hilsdale E, Hugunin J, Kersten M, Palm J, William G. Griswold. An Overview of AspectJ. In: Knudsen JL, ed. Proc. of the European Conf. on Object Oriented Programming. Berlin: Springer-Verlag, 2001. 327–353.

[3] AspectC. 2005. http://www.cs.ubc.ca/labs/spl/projects/Aspectc.html

[4] Spinczyk O, Lohmann D, Urban M. AspectC++: An AOP extension for C++. Software Developer’s Journal, 2005. 68−76.

[5] Spinczyk O, Lohmann D, Urban M. Advances in AOP with AspectC++. In: Hamido F, eds. Proc. of the Software Methodologies, Tools and Techniques (SoMeT 2005). Tokyo: IOS Press, 2005. 33−53.

[6] HyperJ. 2005. http://www.alphaworks.ibm.com/tech/hyperj

[7] AspectR. 2005. http://Aspectr.sourceforge.net/

[8] Kim, H. AspectC#: An AOSD implementation for C#”. Technical Report, TCD-CS-2002-55. Dublin: Trinity College, 2002.

[9] Coady Y, Kiczales G, Feeley M, Hutchinson N, Ong JS. Structuring system Aspects. In: Proc. of the Int’l Conf. on Software Engineering, Aspect-Oriented Programming Workshop. 2001.

[10] Coady Y, Brodsky A, Brodsky D, Pomkoski J, Gudmundson S, Ong JS, Kiczales G. Can AOP support extensibility in client-server architectures? In: Proc. of the European Conf. on Object-Oriented Programming (ECOOP), Aspect-Oriented Programming Workshop. 2001.

[11] Coady Y, Kiczales G, Feeley M, Hutchinson N, Ong JS, Gudmundson S. Position summary: Aspect-Oriented system structure. In: IEEE Computer Society, ed. Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS). Los Alamitos: IEEE Computer Society Press, 2001. 166−167.

[12] Coady Y, Kiczales G, Feeley M, Hutchinson N, Ong JS, Gudmundson S. Exploring an Aspect-oriented approach to OS code. In: Proc. of the 4th ECOOP Workshop on Object-Orientation and Operating Systems. 2001.

[13] Coady Y, Kiczales G, Feeley M, Smolyn G. Using AspectC to improve the modularity of path-specific customization in operating system. In: Proc. of the Joint European Software Engineering Conf. (ESEC) and the 9th ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering (FSE-9). New York: ACM Press, 2001. 88−98.

[14] Mahrenholz D, Spinczyk O, Gal A, Schroder-Preikschat W. An Aspect-oriented implementation of interrupt synchronization in the PURE operating system family. In: Proc. of the 5th ECOOP Workshop on Object Orientation and Operating Systems. 2002. 49−54.

[15] Barreto LP, Douence R, Muller G, S¨udholt M. Programming OS schedulers with domain-specific languages and Aspects: New approaches for OS kernel engineering. In: Coady Y. ed. Proc. of the 1st AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software. 2002. 1−6.

[16] Gal A, Schr¨oder-Preikschat W, Spinczyk O. On minimal overhead operating systems and Aspect-oriented programming. In: ECOOP Workshop on Object Orientation and Operating Systems (ECOOP-OOOSWS 2001). 2001. 1−6.

[17] Gal A, Franz M, Beuche D. Learning from components: Fitting AOP for system software. In: Coady Y, Eide E, Lorenz DH, eds. Proc. of the 2nd AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software. 2003. 43−48.

[18] Stankovic JA, Zhu RQ, Poornalingam R, Lu CY, Yu ZD, Humphrey M, Ellis B. VEST: An Aspect-based composition tool for real-time systems. In: IEEE Computer Society, ed. Proc. of the 9th IEEE Real-Time and Embedded Technique and Applications Symp. Los Alamitos: IEEE Computer Society Press. 2003. 58−69.

[19] Douence R, Fritz T, Loriant N, Menaud JM, Ségura-Devillechaise M, Südholt M. An expressive Aspect language for system applications with Arachne. In: Proc. of the 4th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2005). New York: ACM Press, 2005. 27−38.

[20] Gal A. Reconciliation of an object-oriented runtime environment and resource restricted systems porting PURE to the Atmel AVR [Dip. Thesis]. Magdeburg: Otto-Von-Guericke-Universität. 2001.

[21] Coady Y, Kiczales G. Back to the future: A retroactive study of Aspect evolution in operating system code. In: Proc. of the Int'l Conf. on Aspect-Oriented Software Development. New York: ACM Press, 2003. 50−59.

[22] Åberg RA, Lawall JL, Südholt M, Muller G. Evolving an OS kernel using temporal logic and Aspect-oriented programming. In: Coady Y, Eide E, Lorenz DH, Eds. Proc. of the 2nd AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software. 2003.7−12.

[23] Netinant P, Constantinides C, Elrad T, Fayad M. Supporting Aspectual decomposition in the design of operating systems. In: Proc. of the 14th European Conf. on Object-Oriented Programming (ECOOP 2000), the 3rd Workshop on Object-Orientation and Operating Systems, Sophia Antipolis and Cannes. 2000.

[24] Netinant P, Constantinides C, Elrad T, Fayad M, Bader A. Supporting the design of adaptable operating systems using Aspect-oriented frameworks. In: Arabnia HR, ed. Proc. of the Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000). Las Vegas: CSREA Press, 2000. 271−278.

[25] Lohmann D, Spinczyk O, Schröder-Preikschat W. On the configuration of non-functional properties in operating system product lines. In: Workshop on Aspects, Components and Patterns for Infrastructure Software (AOSD-ACP4IS 2005). 2005. 19−24.

[26] Win BD, Joosen W, Piessens F. AOSD & security: A practical assessment. In: Workshop on Software-Engineering Properties of Languages for Aspect Technologies (AOSD- SPLAT 2003). 2003.

[27] Vanhaute B, Win BD, Decker BD. Building Frameworks in AspectJ. In: Proc. of the Workshop on Advanced Separation of Concerns. 2001. 1−6.

[28] Win BD, Vanhaute B, Decker BD. Security through Aspect-oriented programming. In: Decker BD, Piessens F, Smits J, Herreweghen EV, eds. Advances in Network and Distributed Systems Security. Dordrecht: Kluwer Academic Publishers, 2001. 125−138.

[29] Viega J, Bloch JT, Chandra P. Applying Aspect oriented programming to security. Cutter IT Journal, 2001,14(2):31−39.

[30] Falcarin P, Baldi M, Mazzocchi D. Software tampering detection using AOP and mobile code. In: AOSD Workshop. AOSD Technology for Application-level Security (AOSDSEC). 2004.

[31] Laney R, Linden JV, Thomas P. Evolution of Aspects for legacy system security concerns. In: AOSD Workshop. AOSD Technology for Application-level Security (AOSDSEC). 2004.

[32] Shah V, Hill F. An Aspect-oriented security framework: Lessons learned. In: AOSD Workshop. AOSD Technology for Application-Level Security (AOSDSEC). 2004.

[33] Mahrenholz D, Spinczyk O, Schröder-Preikschat W. Program instrumentation for debugging and monitoring with AspectC++. In: IEEE Computer Society, ed. Proc. of the 5th IEEE Int'l Symp. on Object-Oriented Real-Time Distributed Computing. Washington DC, Los Alamitos: IEEE Computer Society Press, 2002. 249−256.

[34] Lippert M, Lopes CV. A study on exception detection and handling using Aspect-oriented programming. In: Proc. of the 22nd Int'l Conf. on Software engineering. New York: ACM Press, 2000. 418−427.

[35] Manson J, Vitek J, Jagannathan S. Dynamic Aspects for runtime fault determination and recovery. In: Filman RE, Haupt M, Hirschfeld R, eds. Dynamic Aspects Workshop (DAW 2005). Int'l Conf. on Aspect-Oriented Software Development (AOSD 2005). 2005. 27−32.

[36] Filman RE. A bibliography of Aspect-oriented software development (2004). 2005. http://www.riacs.edu/navroot/Research/

[37] 2005. http://aosd.net/

[38] 2005. http://homepages.cwi.nl/~tourwe/ware.html

[39] 2005. http://www.cse.ucsd.edu/users/wgg/Software/AB/

[40] 2005. http://aosd.net/2005/workshops/daw/

[41] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm

附中文参考文献:

[41] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm

陈向群(1961－),女,北京人,教授,CCF 高级会员,主要研究领域为软件工程,操作系统,嵌入式软件.

杨芙清(1932－),女,教授,博士生导师,中国科学院院士,CCF 高级会员,主要研究领域为系统软件,软件工程,软件工业化生产技术.