逆向工程 软件工程 程序理解 演化 (13)

计算机科学 1999Vol. 26№. 5

逆向工程研究与发展

The Research and Development of Reverse Engineering

71-77 TP311.52

袁望洪 陈向葵 谢 涛 郭 耀

(北京大学计算机科学技术系 北京 100871)

Abstract In order to effectively use existing assets in legacy systems, it is important to develop a systematic strategy for the continued evolution of legacy systems. Reengineering offers an approach to migrate a legacy system towards an evolvable system, of which program understanding is a key part. Reverse engineering is effective approach to support program understanding, therefore, plays an important role in successfully reengineering legacy system. This paper is a summarization on reverse engineering. It presents several concepts on reverse engineering, introduces three canonical activities of reverse engineering and a descriptive model of reverse engineering systems that are proposed by Software Engineering Institute at Carnegie Mellon University, and outlines the future directions after introducing several reverse engineering systems.

Keywords Reverse engineering, Program understanding, Reengineering, Legacy system

一、引言

由于多年的运行历史,遗产系统包含了企业的众多知识,包括系统需求、设计决策和业务规则。为了充分有效地利用这些有用资产,遗产系统持续性演化变得十分重要。再工程是将遗产系统转为易演化系统的良好途径之一[22,4]。再工程在检查现有系统基础上,修改系统并组装成新的形式[22,4]。作为工程的问题,再工程涉及对问题的理解,也即,理解系统的当前状态、未来状态和从当前状态演化到未来状态的途径[4]。系统理解基于系统对象、系统专家和系统历史。系统对象包括源代码、手册和运行系统;系统专家包括开发者、维护者和用户;系统历史则包括错误日志和改变记录等。但是,在很多情况下,遗产系统的完整可靠的信息是其程序代码,其他信息必须由此导出[16]。程序理解的目的在于获取足够的系统信息以便让系统按所期望的方式演化,因而,程序理解是遗产系统成功演化的关键之一。

在程序理解中,人们将程序及其环境对应到面向人的概念知识。Brooks 的领域映射理论[1,2,3]把编程过程描述为问题域到实现域的映射构造,程序理解则是逆向构造实现域到问题域的映射,是一个由期待驱动,逐步创建、证实和精化假设的过程。程序理解的效率取决于所理解的程序、理解人员的领域经验和所采用的理解辅助技术。

辅助程序理解的技术有多种,如手工浏览源代码,静态或动态分析目标系统,或者收集和分析系统的度量数据。但这些技术从某种程度上,都可视为逆向工程的支持技术[22]。逆向工程通过标识系统元素、发现元素间关系和产生系统抽象表示,完成下列任务[14]:

· 映射应用程序领域和问题域:计算机程序表示了应用程序领域中的问题。但是,程序通常不包含问题的提示,逆向工程的任务之一是重构应用程序领域到问题域的映射。

· 映射具体和抽象级别:软件开发过程从高层抽象到详细设计再到具体实现。逆向工程则反向运作,从具体细节创建抽象表示。

· 重新发现高层结构:程序是具有良好定义的高层结构的载体。但是,因为时间流失和维护行为,这些高层结构可能已经丢失。逆向工程的任务之一是在程序中检测高层结构。

· 发现程序语法和语义之间丢失的链接:计算机程序是形式化的,具有定义良好的语法和语义。在形式化中,语法正确的程序能够决定输入所对应的输出。但是,有的系统可能已经丢失了原来的语义,

而且有的语言，如面向对象语言，并没有良好的形式化基础。逆向工程要从语法中发现程序语义。

本文是一篇关于逆向工程的综述，介绍了逆向工程的研究与发展。

二、基本概念

"逆向工程"一词来自硬件领域，是通过检查样品开发复杂硬件系统规约的过程[25]，主要指研究他人的系统，发现其工作原理，以达到复制硬件系统的目的。但是，逆向工程也同样适用于其他领域，例如，在软件工程中，逆向工程可用于描述发现自己系统工作原理的过程[16]；而在超文本中，则是指用现有的文本创建联机文档的过程[28]。

在软件领域中，迄今为止还没有逆向工程的标准定义。一般将逆向工程定义为包含两个步骤的过程：第一步分析目标系统，标识系统对象及其关系；第二步创建不同形式或更高抽象层次的系统表示[4]。最近，Scott R Tilley 将抽取和抽象的两个步骤进一步精化为建模、抽取和抽象三个步骤[26]：

建模(model) 采用概念建模技术构造应用程序的领域模型。

抽取(extract) 利用适当的抽取机制从目标系统中收集原始数据。

抽象(abstract) 对目标系统进行抽象，以辅助系统理解并允许浏览、分析和表示抽象结果。

逆向工程适用于软件生存周期的各个阶段和各种抽象层次，包括需求、设计和实现。它可用于低级的抽象层次，例如把程序二进制代码转换为源代码[8]，但主要用于将程序源代码转换为更高抽象层次的表示，如控制流图、数据流图和类关系图等[17,18]。

Chikofsky 和 Cross 在文[4]中定义了其他五个与逆向工程紧密相关的术语：正向工程(forward engineering)、再文档(redocumentation)、设计恢复(design recovery)、重构(restructuring)和再工程(reengineering)。在这些术语体系中，软件系统按生存周期各个阶段分别抽象为需求、设计和实现。依次创建系统抽象的传统开发称为正向工程。再文档和设计恢复是两种形式的逆向工程，再文档是指在同一抽象层次上创建表示，而设计恢复则结合系统的观察和有用的外部信息(如领域知识)标识更高的抽象表示。重构是在保持系统外部行为的前提下改变同一抽象层次上的表示。再工程是通过逆向工程、重构和正向工程的结合把现有系统重组为新的形式。下图显示了它们之间的关系：

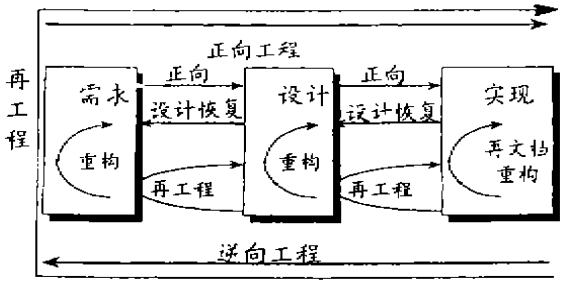

图 1 再工程视图[5]

除了再文档和设计恢复以外，结构化再文档(structural redocumentation)也是逆向工程的重要类型[22]：

再文档 由于种种原因，已有的文档可能不够充分、正确和详细，因而源代码自身成了系统客观可靠的信息源[10]。再文档利用已有的源代码为软件系统逆向生成精确的文档，是逆向工程最老的类型之一[25]。再文档所生成的文档通常是联机文本，但是，也可以产生其他形式的文档，如超文本、交叉索引和图形视图等。

结构化再文档 文档在程序理解中有着重要作用，但是，大多数软件文档都是描述程序的算法和数据结构的小粒度文档，注重于刻画系统的各个部分，很少描述系统的总体结构。对大型遗产系统而言，理解系统的总体结构比理解单个算法更为重要。手工创建从多方面刻画系统总体结构的文档即使可能也是困难的，用逆向工程重建系统总体结构文档的过程称为"结构化再文档"。

设计恢复 文档和结构化文档都是仅仅根据源代码生成文档，因而有着内在的局限性。从设计角度能够更深入地理解系统。设计恢复利用领域知识、外部信息(如规约文档和设计文档)、演绎或模糊推理得到深层抽象[4]，这种抽象的层次比直接检查系统所得的抽象更高。设计恢复能够产生完全理解系统做什么、怎么做以及为什么等所需要的信息，是提高软件构件可复用性的重要途径。

三、规范活动

逆向工程并不改变目标系统，是一个检查的过程，而不是修改的过程。逆向工程通过标识对象、发现其间关系并抽象系统，从而辅助对系统的理解，涉及的对象可分为三类[30]：

· 数据(data)：作为学习、推理和讨论基础的实

际信息。

· 知识(knowledge):所知内容的总和,包括数据以及从数据中推导出的关系和规则。

· 信息(information):相互交织的交流知识。

基于这三类对象,Scott R. Tilley 等人给出了逆向工程的三个规范活动:(1)数据收集,(2)知识组织,和(3)信息浏览[30]。

3.1 数据收集

原始数据是构造和浏览高层抽象的基础,因而数据收集是逆向工程的一项基本活动[20,21,29]。作为构造抽象表示的基础,所收集的数据不能误导或误解,必须客观实际。为此,收集数据时,应尽量遵循下列原则[21,24]:

采用成熟的技术 数据收集所采用的技术包括静态分析、动态分析和获取非正式数据(如调查)等,其中,最为流行的是分析程序源代码,构造带有语法单位及其依赖关系的抽象语法树。在编译领域中,语法分析和交叉索引等技术已经很成熟。采用基于编译的成熟技术,可以得到预期的结果,使收集的数据更加准确和可靠。

利用多种数据 遗产系统有四种数据来源[31]:(1)系统源码,包括数据结构、规则控制进程、报表格式以及计算机的编码指令;(2)系统行为,即系统用户所看到的系统功能和性能;(3)系统文档,包括需求规约、分析文档、设计文档、用户手册和源代码注释;以及(4)系统的设计者和维护者。系统的多种数据从不同方面刻画了系统对象及其关系,从而为深入详细理解系统提供了基础。利用系统源码以外的其他数据,能够更好地辅助软件工程人员理解那些难以理解的系统,如遗产系统。例如,软件工程专家常常利用源代码注释理解系统;调查系统设计者和维护者可以获得他们的经验,从而使数据更准确、更可靠。

过滤数据 为理解大型系统所收集的数据可能是巨大的,以致于超出了人们吸收的能力。人们要理解系统,首先必须吸收其中的数据,要进行必要的删选,而系统理解的关键之一在于了解什么和忽略什么[24]。数据过滤是从丰富的数据源中抽取所选择的对象及其关系,因而,在辅助系统理解中起着重要作用。

3.2 知识组织

对成功的系统理解而言,所收集的数据必须用数据模型保存起来,以便实现有效的存储和检索、辅助对象及其关系的分析,并反映用户对系统特性的了解。数据模型捕捉了系统的本质属性及其关系,可以利用知识组织技术(包括概念建模、领域建模和可扩展知识库)来创建、表示和推理数据模型[1]。

概念建模 传统的数据模型,如层次模型、网络模型和关系模型,都是从适于计算机操作的角度上建模组织数据的。而逆向工程的建模则要以满足人们理解系统为出发点捕捉系统对象及其关系。概念建模(conceptual modeling)强调知识组织,而不是数据组织[1],知识组织是按照系统实体及其语义关系建模的。概念建模所刻画的对象及其关系是面向人的,而不是面向计算机的,其抽象机制能够帮助软件工程人员有效地组织目标系统的知识。因而,概念建模适用于逆向工程领域。

领域建模 当前的逆向工程技术主要以程序分析技术捕捉的程序结构为基础的,但是,程序结构自身并不足以反映程序所应用的问题域。领域分析可视为解决这个问题的有效途径之一。领域分析是标识对象、操作及其在问题域中的联系[7,28]。领域建模是一个标识、组织和表示领域元素及其组成结构的过程[21,29]。它能够帮助人们按领域组织目标系统的知识,识别系统中的标准构件,因而是逆向工程的有效辅助手段[7]。

可扩展知识库 对大型系统而言,逆向工程产生的数据是巨大而复杂的,为了实现有效的存储、检索和分析,有必要用可扩展知识库(scaleable knowledge base)存储逆向工程所得到的知识[21,29]。可扩展知识库并不存储所有的系统对象,而是只存储粗粒度的对象,并支持子系统的增量分析。因而,可扩展知识库能提高大型系统理解的效率。

3.3 信息浏览

大多数程序理解的活动都是在信息浏览时进行的,因而信息浏览可能是逆向工程三个规范活动中最重要的一个。信息浏览通过遍历代表目标系统信息的多维空间,按领域相关的标准分析和过滤信息,并以多种机制表达所得的信息,辅助程序理解中的假设—验证的迭代过程。

遍历(navigation) 对大型系统而言,逆向工程所产生的信息结构并不是线性的,而是一个相互交织的多维信息网。网中的链接代表了逆向工程过程所产生的构件之间的层次关系、继承关系、数据流、控制流和其他关系。遍历采用定向(如地图、多窗口、历史树、路径和复合节点等[18,6])和高级模式匹配等机制,辅助人们浏览逆向工程知识组织阶段所产生的多维信息结构[31]。

分析(analysis) 分析多维信息结构是程序理解的关键。分析从原始数据中推导并抽取那些并不显式存在的信息，并产生关于系统的深层视图。为了辅助用户从多方面理解系统，可利用编程语言机制对分析方法进行编码，从而允许用户根据特定的任务开发特定的分析方法[25]。

表达(presentation) 表达是以可视化方式表示分析的结果。表达是在研究人们对程序理解时所采用的认知策略(cognitive strategy)基础上，创建多方位、正交、结构化视图表现分析结果，以达到一目了然[25]的效果。

四、描述模型

当前，还没有通过逆向工程辅助程序理解的综合途径。相反，有着宽谱系的工具和技术提供对逆向工程三个规范活动的支持，对逆向工程工具和技术的描述也有多种方法。Jim Q. Ning 在其博士论文[15]中标识了逆向工程的四个抽象层次：实现层、结构层、功能层和领域层。实现层描述单个编程单位；结构层描述各个编程单位之间的关系；功能层刻画程序结构与其行为之间的联系；而领域层则刻画应用程序领域相关的概念。

Scott R. Tilley 等人在文[33]中提出逆向工程环境应在以下三个方面具有灵活性：可应用性、可扩展性和自动化级别，并据此给出了逆向工程环境的设计空间[12]，如图 2 所示：

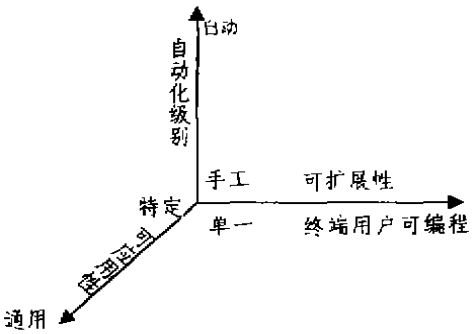

图 2 逆向工程的设计空间[23]

在此基础上，他们还提出了一个描述模型作为逆向工程机制的分类框架[22,30]，该框架从领域可应用性、任务支持和工具集可扩展性三个方面进行描述。

领域可应用性(domain applicability) 描述逆向工程机制是否能应用于某个领域[36]。增大逆向工程机制可用性的途径之一是设计领域特定的工具，这样的系统针对某个特定领域作了裁剪并具有完成特定任务的特殊功能。但是这样的系统不能用于其他领域。而通用的系统又由于性能欠佳和功能缺乏在可用性上受到限制。使逆向工程机制更有力的方法之一是把逆向工程系统设计为领域可重定目标的(domain retargetable)。重定目标系统可经适应性修改后用于多种任务并开发适于当前任务的功能[23]。

任务支持(task support) 描述逆向工程机制是否支持某个特定的任务[30]。逆向工程任务由三个规范活动组合而成，用于完成特定的程序理解目标，譬如模式匹配。在模式匹配中，逆向工程人员通过智力模式识别来标识、操作和浏览目标系统的特定抽象层次上的构件，并将这些构件组装为更抽象的系统表示。逆向工程可以在语法、语义和概念抽象层次上进行模式匹配。编程语言领域中的语法模式匹配称为程序分析；编程语言领域中的语义模式匹配称为计划识别；而应用程序领域中的语义模式匹配称为概念赋值[30,32]。表 1 列出了逆向工程任务、对应的模式抽象层次及其所操作的对象：

表 1 逆向工程任务及其模式抽象层次[30]

任务	模式抽象层次	操作对象
程序分析	编程语言语法	词法单位(token)
计划识别	编程语言语义	计划(plan)
概念赋值	应用程序语义	概念(concept)

工具集可扩展性(toolset extensibility) 描述逆向工程机制是否能扩展支持某个特定的任务[30]。设计人员总是无法预料目标系统的哪些方面对用户重要以及如何向用户表示这些方面。这使得逆向工程工具集要在开放性和封闭性两者之间作出权衡。封闭性系统提供了大量内置功能集，但不能扩展这些功能集；而开放性系统则提供了组装操作和机制用以用户扩展。扩展工具集的途径之一是提供终端用户可定制的功能。通过定制，用户可以在工具集中引入并组装新工具。

五、发展方向

随着从头开发的软件减少，使用的遗产系统相应地增多，逆向工程成为软件工程的研究热点之一。CMU SEI 成立了专门的再工程中心，致力于通过逆向工程进行程序理解技术的标识、增强和实践推广。Chikofsky 等人于 1993 年发起的逆向工程会议每年举行一次，研究和讨论逆向工程的问题、技术及其支

持工具。有关逆向工程的支持系统也很多，典型的有JBPAS[17,18]、DISCOVER[25]和Rigi-[15]等，

JBPAS(青鸟程序分析系统)是一组基于程序分析的工具[17,18]。JBPAS首先通过概念建模建立面向对象程序的概念模型，然后，用编译器前端分析源代码，按照概念模型抽取信息并存入程序信息库中，在信息浏览和使用上，建立一组基于程序信息库的工具，包括设计文档逆向生成工具、测试支持工具、构件提取工具以及度量工具等。JBPAS的前端分析器充分处理了语言的预编译功能，从而准确定位了程序实体与源代码之间的位置关系，并有效地抽取代码注释信息。JBPAS逆向生成的设计文档能为正向工程工具直接使用，从而有效地促进了逆向工程工具和正向工程工具的集成。

DISCOVER是一个管理软件应用程序的开发信息系统，由称为"信息模型(IM)"的库和五个应用程序集(其中的DEVELOP集用于辅助程序理解)组成[25]。信息模型库包含了整个目标系统的依赖关系信息，其建立是在目标系统逻辑和物理结构的映射描述指导下，分析C或C++的源代码抽取信息来完成。DISCOVER通过Gnu编译系统收集数据，并可利用其他工具(如Rational的Quantify[21])产生的信息。IM依赖于知识管理存储并管理整个目标系统的依赖关系信息。DISCOVER信息遍历、分析和表示的功能也相当强大。

Rigi是分析演化软件系统的框架和环境[15]。Rigi用半自动的逆向工程工具从软件表示中抽取数据信息并存储在低层库中，将系统抽象为用分层图形模型刻画的子系统的层次结构，并有一个图形编辑器支持该模型。该图形编辑器提供了编辑、操作、注释和超文本浏览的功能。在用户界面交互时，Rigi用可视化和空间化表示软件信息。可视化表示充分利用了人类视觉系统认知的能力，而空间化表示则允许交互式查询和系统地遍历软件结构。

现有的逆向工程系统为程序理解提供了有效的支持。但是，逆向工程还是一个相当不成熟的领域，理论和实践的研究都还处于早期的探索阶段，有待于在以下几方面进行进一步研究：

1 **注重于经济影响** 逆向工程的目标在于为系统维护和系统演化中的系统理解提供支持，它在实际使用中的经济影响是至关重要的。因而，逆向工程系统必须以实用性为首要目的，要能产生实际的经济效益。逆向工程过程相当复杂，开发全自动的逆向工程工具解决真正的问题在短期内是不可能的。所以，逆向工程工具的研究应注重于实用的半自动工具的开发，并做大量的案例研究[27]。

2 **避免使用虚构的数据** 由于其不成熟，很多逆向工程研究都针对虚构问题，但是，以虚构问题为目标限制了逆向工程系统的实用性。为了使逆向工程的研究真正走向成熟，必须从虚构问题过渡到有经济影响的实际问题。为此，逆向工程研究应当遵循以下几个原则：(1)用实际程序作为例子；(2)用系统作为例子；(3)利用多种信息源[28]。

3. **辅助交流** 逆向工程是相当年轻的研究领域，还没有被广泛接受的标准术语。虽然，Chikofsky等人提出了一组逆向工程相关的术语[4]，但还有待于精化和推广。阻碍交流的另一因素是各个逆向工程原型系统多选用自己的测试数据，从而无法在工具之间作出定量比较。为此，应对某个共同的软件系统演示逆向工程方法及其支持系统，以便作出比较并取长补短。另一方面，逆向工程机制还缺乏统一的分类框架，虽然，Scott R. Tilley等人提出了一个逆向工程描述模型作为分类框架[26]，但是，该框架还有待于在典型逆向工程系统中进行评估和实证研究。

4. **加强Web的利用** 创建能开发和配置工具的基础设施是逆向工程支持机制研究者所面临的挑战之一。现有逆向工程研究和开发的基础设施不够充分，逆向工程系统本身也是昂贵的遗产系统。当前的大多数工具都采用定制的软件系统，重复大量的基本工作(如分析源代码)并且用户界面不统一，难于与其他工具集成。Web提供了解决逆向工程基础设施的良好途径。Web的扩展掩码语言XML[34]可作为表示软件系统构件及其关系的通用中间格式，为查找感兴趣构件、可视化信息空间结构和集成已有工具提供了基础，并且通过统一的用户界面，即Web浏览器，达到表示集成[29]。用Web作为逆向工程的基础设施，还可以有效地支持分布式对象技术带来的分布式程序理解，并利用现有的搜索引擎等技术。

5. **黑盒理解** 传统的逆向工程辅助程序理解的方式可视为"白盒"理解，主要依赖于分析源代码程序抽取程序结构和控制流信息。由分布式对象技术、构件技术和构架技术发展而导致的基于构件的软件开发对程序理解提出了新的挑战。基于构件的系统由已有的构件，尤其是COTS产品和非开发件[6]组装而成，而不是来自于源代码。逆向工程将从传统的白盒理解转向黑盒理解。黑盒理解针对高层

次系统总体结构的理解，偏重于模块之间接口的理解，而不是模块内部的细节[25]。逆向工程的黑盒理解不仅可用于传统的系统维护，而且可用于新构件的验证、适应性修改和组装。很多情况下，系统的成功演化需要结合使用白盒理解和黑盒理解。

参 考 文 献

1 Brooks Ruven. Towards a theory of the cognitive processes in computer programming. Intl. Journal of Man-Machine Studies, 1977, 9: 737～751

2 Brooks Ruven. Using a behavioral theory of program comprehension in software engineering. In: ICSE' 3: Proceedings of the 3rd Intl. Conference on Software Engineering. Atlanta, Georgia, May, 1978. 10～12

3 Brooks Ruven. Towards a theory of the comprehension of computer programs. International Journal of Man-Machine Studies, 1983, 18(6): 543～554

4 Chikofsky Elliot J, Cross II James H. Reverse engineering and design recovery: A taxonomy. IEEE Software, 1990, 7(1): 13～17

5 Cifuentes Cristina, Gough K John. Decompilation of binary programs. Software-Practice and Experience, 1995, 25(7): 811～829

6 Casanova M A, et al. The Nested Context Model for Hyperdocuments. In: Hypertext' 91, San Antonio, TX, 1991

7 DeBaud J-M, et al. Domain Analysis and Reverse Engineering. In: Intl. Conference on Software Maintenance, Victoria, British Columbia, Canada, 1994

8 Foreman John, et al. Software Technology Review, Draft. Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, June, 1997

9 Feiler Peter H. Reengineering, An engineering problem. [Technical Report CMU/SEI-93-SR-5] Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, July 1993

10 Fletton Nigel T, Munro Malcolm. Redocumenting software systems using hypertext technology. In: Proceedings of the 1988 Conference on Software Maintenance (CSM' 88). Phoenix, Arizona, October, 1988. 24～27

11 Kristensen B B, Osterbye K. Conceptual Modeling and Programming Languages. ACM SIGPLAN Notices, 1994, 29

12 Lane T G. Studying software architecture through design space and rules: [Technical Report, CMU/SEI-90-TR-18]. Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, 1990

13 Layzell P. Document and Code Knowledge Elicitation Tool Set, ESPRIT Project Proposal, UMIST, Manchester, G. B, 1990

14 Mohammad Ashrafuzzaman Homepage. RE [online]. Available at, WWW URL: http://www.cs.usask.ca/homepages/grads/moal135/856/RE/(1996)

15 Mylopoulos John, et al. Towards an integrated toolset for program understanding. In Proceedings of the 1994 IBM CAS Conference (CASCON'94). Toronto, ON, 1994. 19～31

16 Muller Hausi A, et al. Understanding software systems using reverse engineering technology: Perspectives from the Rigi project. In: Proceedings of the 1993 IBM/NRC CAS Conference (CASCON'93) Toronto, Ontario, October, 1993. 25～28

17 Mei Hong, et al. BDCom-C++: A C++ Program Understanding System. Chinese Journal of Electronics, 1997, 6(2): 64～69

18 Nielsen J. Hypertext and Hypermedia Academic Press, 1990

19 Ning Jim Q. A Knowledge-Based Approach to Automatic Program Analysis: [PhD thesis]. Department of Computer Science, University of Illinois at Urbana-Champaign, 1989

20 Parnas D L. Designing software for ease of extension and contraction. IEEE Transactions on Software Engineering, 1979, SE-5(2): 128～137

21 Rational Software Corp. Quantify [online]. Available at: WWW URL: http://www.rational.com/products/quantify/(1998)

22 Reengineering Center, Perspectives on Legacy System Reengineering, Draft Version 0.3. Pittsburgh, PA Software Engineering Institute, Carnegie Mellon University, December 1996

23 Rekoff M. On reverse engineering, IEEE Transactions on Systems, Man and Cybernetics, 1985, 3-4: 244～252

24 Shaw M. Larger Scale Systems Require Higher-Level Abstractions. ACM SIGSOFT Software Engineering Notes, 1989, 14: 143～146

25 Sneed H M. Software renewal: A case study IEEE Software, 1984, 1(3): 56～63

26 Software Emancipation, The DISCOVER Development Information System (Version 4.0) [online]. Available at: WWW URL: http://www.setech.com(1996)

27 Selfridge Peter G, et al. Challenges to the field of reverse engineering. In: Proceedings of the 1993 Working

Conference on Reverse Engineering(WCRE'93) Baltimore, Maryland, May, 1993. 21～23

28 Tilley Scott R. Domain-Retargetable Reverse Engineering. [PhD thesis] Department of Computer Science, University of Victoria, 1995. Available as technical report DCS 234-IR

29 Tilley Scott R. Coming Attractions in Program Understanding I. Highlights of 1997 and Opportunities in 1998. [Technical Report, CMU/SEI-98-TR-001]. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, February, 1998

30 Tilley S R, et al. Towards a Framework for Program Understanding. In: Proceedings of the 4th Workshop on Program Comprehension. Berlin, Germany, 1996. 29～31

31 Tilley Scott R, Smith Dennis B. Coming Attractions in Program Understanding: [Technical Report, CMU/SEI-96-TR-019]. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, December, 1996

32 Tilley Scott R, et al. Personalized information structures. In: Proceeding of the 11th Annual International Conference on Systems Documentation (SIGDOC'93). Waterloo, Ontario, October, 1993. 5～8

33 Tilley Scott R, et al. Programmable Reverse Engineering. Intern. J. of Software Engineering and Knowledge Engineering, 1994, 4(4): 501～520

34 The World Wide Web Consortium. Extensible Markup Language (XML) [online]. Available at: WWW URL: http://www.w3.org/XML/(1998)

35 Wasserman A I. Tool integration in software engineering environments. In: Goos G, Hartmanis J, eds. Proceedings of the International Workshop on Environments. Chinon, France, September, 1989. 18～20

36 Wu Qiong et al. BDCom-ST: A Smalltalk program understanding system. Journal of Software, 1996(5)

37 Yang Fuqing et al. Experiences Writing C++ Compiler Front End. SIGPLAN Notices, 1998(3)

38 Yuan Wanghong, et al. C++ Program Information Database for Analysis Tools. Software-Practice and Experience, 1998(4)

《计算机科学》参考文献表的编写格式

《计算机科学》采用顺序编码制：在文后参考文献表中，各条文献按在论文中的文献序号排列顺序，项目应完整，内容、著录格式与符号应正确。所用参考文献应列出主要者，尽量限制在10篇内。下面列出常见的几种编写格式：

1 专著

标引项顺序号 著者．书名．版本．其他责任者．出版地：出版者，出版年

2 论文集中析出的文献

标引项顺序号 作者．题名．见(In)：编者．文集名．出版地：出版者，出版年．在原文献中的位置

3 期刊中析出的文献

标引项顺序号 作者．题名．其他责任者．刊名，年，卷(期)：在原文献中的位置

4 学位论文

标引项顺序号 作者．题名：[学位论文]．保存地：保存者，年份

在参考文献表中，用各种文字书写的姓名，一律姓在前名在后；外国人名可缩写为首字母(大写)，但不加缩写点。