软件复杂性研究综述①

姜 茸[1,2,3], 杨 明[2]

[1](云南财经大学 信息学院, 昆明 650221)
[2](云南大学 软件学院, 昆明 650091)
[3](云南大学 发展研究院, 昆明 650091)

摘 要: “软件危机”的根源在于软件的复杂性, 当复杂性超过人们能够控制的程度时, 软件项目的失败便是必然. 系统地介绍和归纳总结了国内外软件复杂性研究现状, 分析了目前研究存在的问题, 对未来软件复杂性研究热点方向进行了探讨与展望.

关键词: 软件; 复杂性; 综述

Survey on Software Complexity Research

JIANG Rong[1,2,3], YANG Ming[2]

[1](School of Information, Yunnan University of Finance and Economics, Kunming 650221, China)
[2](School of Software, Yunnan University, Kunming 650091, China)
[3](School of Development Studies, Yunnan University, Kunming 650091, China)

Abstract: The software complexity is the origin of Software Crisis, and a software project must be failure when the complexity is higher than the controllable degree of the manager. In this paper, an induction and summarization of complexity research actuality is given systematically at first. Then, the paper analyzes the problems existed in the current research, also discusses and prospects the future of software complexity research.

Key words: software; complexity; survey

为摆脱“软件危机”, 北大西洋公约组织成员国软件领域专家于 1968 年和 1969 年两次召开 NATO 会议, 并提出软件工程概念, 从此软件工程成为学者们研究的热点之一. 生成高质量的软件是所有的软件工程方法努力实现的目标, 然而, 软件系统的设计开发却总不能尽如人意, 半个世纪之后的今天, “软件危机”依然没有得到很好地解决, 软件项目失败事件仍时有发生. 据《CIO INSIGHT》杂志报导, Standish Group International 收集并公布全世界 IT 项目的有关信息, 其 CHAOS 研究报告显示, IT 项目成功率 1994 年 16%, 2006 年 34%; 2008 年 32%, 大概有 52.7%的 IT 项目花费是原计划的 189%. 美国 Gartner Group 公司于 2000 年从北美 1375 份资料中调查显示, IT 项目中有 40%是失败的. Price Waterhouse 2004 年的研究也表明, 全世界各种类型的软件项目中, 有一半是失败的. 这些可以量化的数据让人触目惊心, 项目失败的最主要原因是软件系统的复杂性越来越高.

越复杂的软件就越难以保证其质量、费用和生产率, 当其复杂性超过人们能够控制的程度时, 该软件项目的失败便是必然. 只有复杂性在可控范围内, 软件项目才能成功. 复杂性是一个贯穿整个软件生命周期的重要因素, 它和软件生产的成本、质量、生产和价格密切相关. 业界学者和实践者都一致公认复杂性是使软件系统出错、难于理解和产生安全问题的重要

① 基金项目:国家自然科学基金(61263022,61303234);中国博士后科学基金(2012M521722);国家社会科学基金(12XTQ012);教育部人文社会科学研究青年基金项目(11YJCZH073);云南省哲学社会科学规划项目(QN201210);云南省自然科学基金(2010ZC100)
收稿时间:2013-12-27;收到修改稿时间:2014-03-18

原因. 因此, 研究软件复杂性具有十分重要的意义.

1 复杂性与软件复杂性

复杂与简单相对, 越复杂则越难理解、管理和控制, 这是人们对复杂性的普遍认识.

复杂性问题于上世纪 40 年代末由系统科学先驱贝塔朗菲提出, 他预见到系统科学本质上是研究复杂性的科学. 信息论创始人之一韦弗在同一时期提出有组织复杂性和无组织复杂性的划分, 把有组织复杂性作为系统科学的研究对象. 但总的来说, 那个时期的复杂性研究尚无实质性进展. 目前复杂性研究已经取得一些成果, 但究竟什么叫复杂性仍无统一定义. 按照传统的理解, 一个事物在未被认识之前是复杂的, 一旦被认识就成为简单的了. 从人类认识事物的过程看, 这种情形是常见的. 但现代科学技术的发展表明, 不能把复杂性全部归结为认识过程的不充分性, 必须承认存在客观的复杂性, 真正的复杂性应当具备自身特有的规定性, 即便已被人们认识, 即使找到解决办法, 它仍然是复杂的. 上世纪 90 年代, 钱学森提出:

凡是不能用还原论方法处理的或不宜用还原论方法处理的问题, 而要用或宜用新的科学方法处理的问题, 都是复杂性问题.

软件复杂性是复杂性研究领域的一个子问题, H. Zuse 将它定义为:

软件复杂性的真正含义就是分析、设计、测试、维护、改变和理解软件的困难[1].

随着软件复杂程度的增加, 对软件复杂性的研究也变得越来越重要, 它对开发费用的合理安排, 开发周期的预测以及估计错误数等方面起着很重要的参考作用.

2 国内外研究现状

目前国内外软件复杂性研究主要涉及的领域如图 1 所示.

2.1 程序复杂性

程序复杂性通常是指关于软件规模、开发难度以及软件内部可能隐藏错误多少的一种抽象表示, 软件规模及难度与软件报价、完成周期有关, 是生产招投标的依据. 图 1 中程序复杂性度量具有最早的研究历史, 成果也很多.

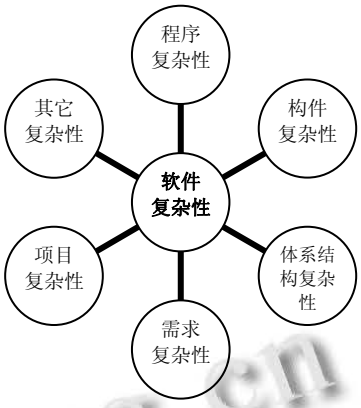

图 1 软件复杂性研究框架

①程序复杂性经典度量

程序复杂性度量大致可分为面向过程的结构化程序复杂性度量和面向对象程序复杂性度量两个方面, 较经典的度量方法如表 1、表 2 所示.

表 1 结构化程序复杂性度量方法

名称	简介
LOC 代码行度量	代码行数度量法以程序的总代码行数作为程序复杂性的度量值, 一般用于估算程序开发的工作量. 一般来说代码行数越多程序也越复杂. 这种方法的主要问题是度量不够精确、不同语言不同类型语句的复杂性也不同.
程序控制结构复杂性度量	1976 年, McCabe 提出了基于程序拓扑结构的复杂性度量模型[2], 通过计算强连通的程序图中线性无关有向环的个数来进行程序复杂性的度量.
程序文本复杂性度量	1977 年, Halstead 从统计学和心理学的角度研究程序复杂性问题, 提出了一种定义和度量软件复杂性的理论, 即 Halstead 度量法[3], 它是通过计算程序中运算符和操作符的数量来对程序的复杂性加以度量.
Harrison 复杂性度量	1981 年, Harrison 提出了一种将流程图分解为行列为基础的程序复杂性度量方法[4]. 按照 Harrison 等人的观点, 可确定在结构化, 特别是非结构化的流程图中的节点嵌套层次.

表 2 面向对象程序复杂性度量方法

名称	简介
C&K 度量	1994 年, Chidamber 和 Kemerer 奠定了面向对象设计中的软件度量学理论基础, 提出基于继承树的 C&K 度量方法[5, 6]. C&K 度量由六条适用于面向对象设计的度量准则组成, 分别是度量每个类中所有方法的复杂度 WMC、继承深度 DIT、子孙个数 NOC、一个类的响应个数 RFC、对象间耦合程度 CBO 和方法中内聚缺乏度 LCOM.
MOOD 度量	1995 年, Abreu 等人提出的 MOOD 度量[6, 7], 从

	封装性、继承性、耦合性和多态性四个方面给出了面向对象软件六个度量指标, 分别是方法隐蔽因子 MHF、属性隐蔽因子 AHF、方法继承因子 MIF、属性继承因子 AIF、耦合因子 CF 和多态因子 PF.
Chen&Liu 度量	1993 年, J-Y Chen 和 J-F Liu 从八个方面给出了软件度量方法[6, 8], 分别是操作复杂度、操作参数复杂度、属性复杂度、操作耦合度、类继承度、内聚度、类耦合度和重用度.

②其它结构化程序复杂性研究

文献[9]介绍了一个基于随机单值响应函数熵评估和软件特征函数的结构化程序复杂性计算方法. 文献[10]将软件的复杂性视为程序的复杂性, 提出嵌入式应用软件复杂性估计通用平台的概念. 文献[11]基于 McCabe 和 Halstead 两种结构化程序复杂性度量方法, 介绍了一个程序复杂性度量系统.

③其它面向对象程序复杂性研究

维护阶段的一个重要活动是首先理解源代码, 然后修改它, 这需要整合并理解数据使用和类方法, 文献[12]基于类和对象, 介绍了两种面向对象软件空间复杂性的度量方法. 文献[13]从相似系统学出发研究了面向对象软件系统的复杂性测度问题, 提出了对象关联复杂度计算公式. 文献[14]基于最小事件通讯组介绍了通讯复杂性单元的概念, 并提供了一个通用的软件复杂性度量方法用于度量分布式程序的维护复杂性. 文献[15]提出了一个面向对象系统的软件复杂性模型, 该模型从变量、方法、对象和系统层次定义了面向对象系统的软件复杂性. 在每一个层次, 度量被确定用于解释系统的内聚和耦合方面. 文献[16]用 Z 规格面向对象软件复杂性度量, 该文提出一个基于层次的继承度量方法, 用于度量面向对象软件继承层次复杂性. 文献[6]通过继承图描述面向对象软件复杂性度量方法. 提出了单元重复继承算法生成单元重复继承. 该方法未涉及面向对象技术对软件内部属性的变化.

④其它程序复杂性研究

文献[17]认为程序复杂性因素分为: 语句、功能、计算复杂性; 每一个因素均从长度、时间、层次或深度当中的某一方面表示复杂性因素的数量; 用这些思想, 基于离散数学程序结构, 该文定义了复杂性因素的有序度量. 为了维护软件, 程序员需要理解源代码; 文献[18]研究程序代码和数据空间复杂性, 这关系到软件的易懂性; 该文中程序复杂性主要指结构化程序, 也兼有面向对象程序.

2.2 构件复杂性

构件是软件系统中具有相对独立功能和结构, 可以明确辨识、接口由契约指定、语境有明显依赖关系、可独立部署、可组装的软件实体元素, 并且可以重复使用. 在传统结构化软件的复杂性研究中, 更多地关注模块内部的细节, 而构件软件更多地关注构件及构件之间交互关系. 文献[19]提出了一种基于依赖矩阵的构件软件复杂性度量模型.

2.3 体系结构复杂性

软件体系结构是具有一定形式的结构化元素, 即构件的集合, 包括处理构件、数据构件和连接构件. 软件体系结构不仅指定了系统的组织结构和拓扑结构, 并且显示了系统需求和构成系统的元素之间的对应关系, 提供了一些设计决策的基本原理, 已经成为软件工程一个重要研究领域. 过于复杂的体系结构会带来较低的可理解性和可维护性, 过于简洁的体系结构又有可能造成可靠性和安全性下降. 文献[20]提出一种软件体系结构复杂性的评价方法, 通过抽取构件之间的连接, 提出了基于复合关系的分层式软件体系结构, 在该结构基础上, 提出了基于加权图的复杂性评价模型. 文献[21]利用代数表达式对软件体系结构的直观复杂性进行了度量. 文献[22]探讨了基于复杂网络的软件结构复杂性度量, 具体涉及网络节点复杂性、交互复杂性、聚集复杂性.

2.4 需求复杂性

需求分析是软件过程中非常重要的一个环节, 需求分析失败则一切都失败. 文献[23]以面向对象的分析方法为基础, 在需求陈述模板下, 提出了软件需求阶段的复杂性定量度量和需求陈述有效性检验方法. 文献[24]在量化用例的基础上研究用例复杂度.

2.5 软件项目复杂性

软件项目复杂性是软件项目管理中很重要的指标, 文献[25]提出了基于证据推理的软件项目复杂性评估模型和方法.

2.6 其它复杂性

文献[26]从复杂性科学的角度剖析了软件开发系统及其开发过程复杂性产生的原因, 认为软件开发的复杂性主要来源于开发系统的复杂性和在开发系统框架内实施开发过程的复杂性, 人是复杂性的主要根源. 原型用于查找和矫正缺陷, 并确保没有新的缺陷引入

到软件过程的开发阶段；原型开发旨在提高软件质量、降低成本，以及修正、增强和精化软件；文献[27]着重于第二方面，设计了一个新的开发模型及相关软件复杂性度量方法，研究软件维护环境中合成复杂性. 文献[28]研究认知复杂性对项目领导业绩的影响，报道了一个初步实验测试的结果，该实验测试了认知差别和集成认知复杂性对于项目领导的重要性，并宣称实验结果表明认知复杂性对于项目领导的成功是重要的.

3 存在的问题及未来研究方向

3.1 存在的问题

①对软件复杂性理解有误

“软件是程序、数据和相关文档的集合”这是人们对软件的公认定义，但由于早期软件开发几乎仅仅只是程序设计，故许多学者认为软件复杂性就是程序复杂性，这样理解的学者和文献不胜枚举.

②复杂性研究内容边界模糊

软件复杂性研究已有 40 多年的历史，但软件复杂性具体包括哪些方面？鲜有文献回答，笔者仅发现文献[26]涉足此问题，它将软件开发的复杂性归纳为：结构复杂性、边界复杂性、演化复杂性、概念多元性、认识有限性与多重偶然性. 文献[1]虽给出了软件复杂性的定义，但对此问题的认识也是模糊的. 按软件的定义，软件复杂性除了程序复杂性之外，至少还应该包括文档复杂性.

③“微观”研究有余、“宏观”研究不足

结构化程序复杂性研究起步最早，主要关注程序模块内部的细节. 面向对象的程序复杂性研究主要聚焦于类的内部复杂性(如类的加权方法数、类内聚度、方法隐藏因子、属性隐藏因子)、类之间继承复杂性(如继承深度、继承方法数比例等)、类之间的耦合复杂性(如对象类之间的耦合数、所有可能的关联类对的数目等). 这些研究无一例外的都是程序的微观复杂性研究，缺乏全局和总体上的复杂性研究，仍停留在程序内部的微观属性上.

软件构件、软件体系结构、需求分析、整个软件项目相对程序均属于宏观研究. 基于构件的软件开发技术是当前软件开发的世界潮流，是现代软件工业化生产必须采用的方法，但是这方面的研究鲜见报导. 大量实践统计表明：大软件系统开发中 70%的错误是由需求和软件设计阶段引入的，可见需求和设计阶段的重要性，需求错误会给后期带来许多麻烦，甚至全盘皆输，需求越复杂则越容易出错，因此需求复杂性研究具有重要的现实意义，但是人们对此领域的研究不多，尤其是定量的复杂性分析. 软件体系结构试图在软件需求与软件设计之间架起一座桥梁，着重解决软件系统的结构和需求向实现平坦地过渡的问题，是软件开发生命周期所有活动的蓝图，它已成为软件工程领域重要的研究热点之一. 但是，软件体系结构复杂性的研究成果不多. 软件项目与软件既有区别也有联系，软件是软件项目的最终产品，也是最重要的产品，笔者认为软件复杂性应该包括软件项目复杂性，但这方面的研究还很少.

④技术因素复杂性研究较多、非技术因素的复杂性研究较少

纵观软件复杂性研究历程，程序复杂性研究最为成熟，成果最多，这显然属于技术因素复杂性. 构件和体系结构复杂性也应归为技术因素复杂性，研究方兴未艾. 业界有一种说法认为，非技术因素往往更容易导致软件项目失败. 软件由程序和文档组成，由于文档的复杂性，它涉及太多非技术因素，目前研究比较少. 复杂性的一个显著特征是不确定性，软件项目的需求几乎都是不确定的，这也难怪软件项目的失败率总是比其它项目高，比如工程建筑类项目. 之外，糟糕的项目定义、项目计划、不现实的预期以及沟通不畅、需求变更管理不当、较低的用户参与等，这些都是导致失败的非技术因素，这些问题可以归为需求工程和软件项目管理，研究这方面复杂性的学者不多.

3.2 未来研究方向

目前，无论是面向过程还是面向对象的程序复杂性研究均趋于成熟，这方面研究不会再有突破，笔者预计未来软件复杂性研究热点方向应是：

①软件构件和软件体系结构复杂性

基于构件和体系结构的软件开发方法是大势所趋，其复杂性研究具有重要的理论和实际意义. 目前，软件构件和软件体系结构复杂性研究方兴未艾，它必将是未来软件复杂性研究的热点之一.

②软件项目复杂性

现代软件开发一般都是小组来完成，一个人开发的历史已经不复存在. 所以，技术不是关键，好的管理才是开发出好软件的前提. 软件开发已不能像从前

那样把技术放在第一, 而是该把管理放在第一位. 一个成功的软件不一定拥有最好的技术, 但在它背后一定有一个好的管理. 软件项目复杂性的高低与项目可控程度成反比, 研究项目复杂性可为项目管理和决策提供重要依据和参考.

③需求复杂性

需求分析与管理属于软件项目研究的子领域, 由于它对项目成败至关重要, 这里将它与软件项目复杂性分开单独列出. 在软件项目的开发过程中, 需求变更贯穿了软件项目的整个生命周期. 需求的复杂性主要体现在: 需求描述、需求完备程度、需求细致程度、需求变化等方面, 目前主要是一些定性的研究, 定量化的需求复杂性分析将是未来软件复杂性研究的重要方向.

4 结语

软件复杂性与软件质量、成本和生产进度均直接相关, 与可控制性成反比, 它是软件工程领域中非常值得深入研究的一个方向. 从 LOC 程序复杂性度量至今, 软件复杂性研究已经取得许多重要成果. 本文系统地综述了国内外研究现状, 鉴于篇幅所限对前人研究成果未作详细阐述, 仅作简要介绍, 感兴趣的读者请参阅相关文献. 之外, 分析了现有软件复杂性研究存在的问题, 并对未来研究热点方向进行预测. 软件复杂性研究要形成完整的理论体系, 首先必须明确定义, 明确研究内容框架; 本文提出了此问题, 该问题的解决和软件项目复杂性将是笔者下一步的研究内容.

参考文献

1 Zuse H. Software Complexity Measures and Models. 1990, New York, NY: de Gruyter and Co.

2 McCabe. A complexity measurement. IEEE Trans. on Software Engineering, 1976, 2(4): 302–308.

3 Halstead. Elements of Software Science. 1977, New York: Elsevier North-Holland.

4 Harrison W, Kenneth M. A complexity measure based on nesting level. ACM SIGPLAN Notices. ACM SIGPLAN Notices, 1981, 6(3): 63–74.

5 Chidamber SKC. Metrics suite for object oriented design. IEEE Trans. on Software Engineering, 1994, 20(6): 476–493.

6 伦立军,丁雪梅,李英梅,张翼.基于继承图的面向对象软件复杂性度量研究.计算机工程与应用,2006,27:93–95.

7 Brito FAE. MOOD-metric for object-oriented design. Internaltional Workshop on Pragmatic and the Oretical Directions in Object-Oriented Software Metric. Portlan. OOPSLA'94. 1994.

8 Chen JY, et al. A new metric for objec oriented design. Information and Software Technology, 1993, 35(4).

9 Roca JL. An entropy-based method for computing software structural complexity. Microelectron Reliab, 1996, 36(5): 609–620.

10 徐翥,郦萌.应用软件复杂性通用估计平台实现可行性研究.同济大学学报,2004,32(4):529–533.

11 夏红霞,童维农,邹承明,鄂勇辉,钟珞.软件复杂性度量系统的研制.计算机应用,2000,(4):16–18.

12 Jitender Kumar Chhabra KKA, Singh Y. Measurement of object-oriented software spatial complexity. Information and Software Technology, 2004(46): 689–699.

13 雷战波,许倩钰.基于相似学的面向对象软件复杂性的定量测度研究.计算机工程与科学,2002,24(1):93–96.

14 Tsaur WJ, S.-J.H. A new generalized software complexity metric for distributed programs. Information and Software Technology, 1998, 40: 259–269.

15 Tegarden DP, Monarchi DE. A software complexity model of object-oriented systems. Decision Support Systems, 1995, 13: 241–262.

16 Shih TK, Chung CM. Using Z to specify object-oriented software complexity measures. Information and Software Technology, 1997, 39: 515–529.

17 Gonzalez RR. A unified metric of software complexity: Measuring productivity, quality, and value. Journal of Systems Software, 1995, 29:17–37.

18 Chhabra JK, Singh Y. Code and data spatial complexity: two important software understandability measures. Information and Software Technology, 2003, 45: 539–546.

19 焦锋,王丽平,侯建民.基于依赖矩阵的构件软件复杂性的度量模型.计算机应用与软件,2009(5):55–56.

20 焦锋,王丽平,侯建民.一种软件体系结构复杂性的评价方法.计算机应用研究,2008,(8):2377–2379.

21 黄万艮,陈松乔.基于构件运算的软件体系结构及其复杂性度量.计算机工程与应用,2007,43(14):66–70.

22 李兵,王浩,李增扬,何克清,余敦辉.基于复杂网络的软件复杂性度量研究.电子学报,2006,(12):2371–2375.

23 张遂征.软件需求复杂性度量与有效性检验方法.中国铁道科学,2002,(12):30–37.

24 王悠,张熙.用例驱动的软件复杂性度量及应用.计算机工程与设计,2007,28(11):2543–2546.

25 蒋国萍,陈英武.基于证据推理的软件项目复杂性评估.计算机工程与应用,2005,(2):4–7.

26 林正奎,杨德礼,杨红.软件开发系统六元结构模型及其复杂性研究.系统工程学报,2006,21(4):368–374.

27 Hops JM, et al. Development and application of composite complexity models and a relative complexity metric in a software maintenance environment. Journal of Systems Software, 1995, 31: 157–169.

28 Green GC. The impact of cognitive complexity on project leadership performance. Information and Software Technology, 2004, 46: 165–172.