中国科学 F辑: 信息科学 2009 年 第 39 卷 第 9 期: 946～950
www.scichina.com info.scichina.com
《中国科学》杂志社
SCIENCE IN CHINA PRESS

软件可信性动力学特征及其演化复杂性

郑志明①③*, 马世龙②*, 李未②, 韦卫③, 姜鑫①, 张占利③, 郭炳晖①

① 北京航空航天大学数学、信息与行为教育部重点实验室, 北京 100191
② 北京航空航天大学软件开发环境国家重点实验室, 北京 100191
③ 北京大学数学科学学院, 北京 100871
*通信作者 E-mail: zzheng@pku.edu.cn, slma@nlsde.buaa.edu.cn

收稿日期: 2008-08-01; 接受日期: 2008-10-10
973 国家重点基础研究发展计划 (批准号: 2005CB321900)、国家自然科学基金 (批准号: 60473019) 资助项目

摘要 可信软件已成为现代软件技术发展和应用的重要趋势和必然选择, 而软件可信性建模已成为构造可信软件的先决条件和必要手段. 为了探讨和阐明软件可信性的基本科学问题、建立软件可信性度量的理论基础, 文中结合动力系统的基本思想探讨软件可信性及其演化规律, 研究在各种内部和外部因素作用下软件可信性演化的动力学机制, 并建立相应的动力学模型, 从而软件系统的可信性可以认为是软件系统在动态开放环境下其行为的统计特性. 通过对两个简单实例的建模分析, 说明了软件系统可信属性的极限演化行为与动力系统特征的对应关系, 诠释了软件可信性的动力学特征及其演化复杂性.

关键词
软件可信性
可信特征属性
演化复杂性
动力系统

0 引言

最近十多年来, 软件系统的开发和运行环境已经从静态封闭转化为动态开放. 运行环境开放多变导致了多种不确定性因素, 使得软件系统的正确性、安全性、可靠性、可用性和实效性等性质难以保证, 甚至软件系统本身的生存性都难以保证. 在开放多变的环境中, 软件系统变得越来越脆弱, 让人们感觉难以信任, 很多时候不以人们期望的方式工作, 发生各种故障和失效, 直接或间接地引发事故甚至灾难. 因此, 软件系统的可信性成为当前国内外学术界和产业界非常关注和重点投入的研究课题.

软件系统提供的服务是否可信, 特别是在动态、开放和分布式环境下应用软件系统是否可信, 是 "安全攸关 (safety-crucial) 应用领域"(例如, 金融应用软件、航天应用软件等) 中的至关重要问题. 如何在软件的设计、开发、运行和维护中保证软件系统具有可信性是当前软件系统可信性研究领域的重要内容.

1999 年, IBM、HP、Intel、微软等著名 IT 企业发起成立了可信计算平台联盟 TCPA(trusted computing platform alliance), 2003 年 TCPA 改组为可信计算组织 TCG. TCPA 和 TCG 的出现形成了可信计算的新高潮. 该组织不仅考虑信息的秘密性, 更强调信息的真实性和完整性, 而且更加产业化和更具广泛性. 微软提出了名为 Palladium(后改名为 NGSCB) 的可信计算计划, 此后又推出支持可信计算机制的新一代操作系统 VISTA. Intel 为支持微软的 Palladium 计划宣布 LaGrande 硬件技术, 并计划推出相应的新一代处理器. 欧洲于 2006 年 1 月启动了名为 "开放式可信计算 (Open Trusted Computing)" 的研究计划 [1].

虽然软件系统的可靠性得到了一定程度的发展, 但直至今日, 关于可信性仍然没有统一的认识和说法. 不过可信性的理念已经深入到计算机硬件、计算机软件和计算机网络的各方面. 可信计算组织 TCG

引用格式: 郑志明, 马世龙, 李未, 等. 软件可信性动力学特征及其演化复杂性. 中国科学 F 辑: 信息科学, 2009, 39(9): 946–950

用实体行为的预期性来定义可信: 一个实体是可信的, 如果它的行为总是以预期的方式, 达到预期的目标. ISO/IEC 15408 标准将可信定义为: 参与计算的组件、操作或过程在任意的条件下是可预测的, 并能够抵御病毒和物理干扰. 从用户角度看, 所谓可信是指计算机系统所提供的服务是可信赖的, 而且这种可信赖是可论证的. 基于上述说法, 一些学术文献中将可信性看作是安全性或可靠性 [2~5], 另外一些学术文献则认为可信性应该包括更广泛的内容, 例如, 应包括保密安全性、可用性、容错性和生存性 [6~11]. 还有学者认为, 可信性是软件系统的质量属性, 应该包括正确性、可靠安全性、保密安全性和可用性 [12].

目前, 在软件可信性领域, 理论研究相对滞后, 主要研究方法处于技术手段, 文献 [13] 曾提出用物理学方法研究互联网环境下海量信息组织和管理软件系统的行为的观点, 包括用数学物理方程描述海量信息存储和传输遵从的物理学规律, 并通过模拟仿真和实验进行验证. 但在这方面尚缺少系统深入的科学研究, 缺少对软件可信性基本科学问题的认识和探讨. 可信计算的技术实践已经取得长足的发展, 有必要在可信软件的实践中丰富和发展软性可信性理论, 包括软件可信复杂性、软件可信性度量模型和评估方法与准则. 可以认为, 软件系统的可信性是软件系统在动态开放环境下其行为的统计特性. 因此, 软件系统可信是指: 软件系统的行为及其结果是可预期的、行为状态是可监测的、行为结果是可评估的、行为异常是可控的.

针对以上问题并综合考虑软件可信性的各种相关属性, 在对软件系统可信性的演化形式分析的基础上, 本文综合考虑软件可信性内外部因素相互作用, 建立软件可信性的动态演化模型, 并对可信性的演化复杂性进行研究分析, 论证软件系统可信属性的极限演化行为与动力系统特征的对应关系, 诠释软件系统的行为特性及其可信复杂性, 给出其演化复杂性的分析方法, 旨在为软件可信性的判定提供研究方法和技术手段.

1 软件可信性演化的 3 种形式及其动力学特征

一般地, 软件系统的可信性包括: 身份可信、功能可信和行为可信, 其中行为可信涉及到软件系统的行为演化过程, 是软件系统可信性的核心问题. 本文将基于动力系统理论和方法重点研究软件系统的行为可信性, 旨在建立软件系统 (行为) 可信性演化的动力学模型, 进而提出基于动力系统理论的软件可信性分析、建模与预测方法.

软件系统可信性特征属性包括局部特征属性和全局特征属性, 例如: 可用性、可靠性、可靠安全性、容错性、实时性、一致性、可测性等 [12]. 开放复杂环境下的软件作为一个在其生命周期内始终处于动态的系统, 在内部和外部因素的作用下不断演化. 软件在复杂变化环境中的这种动态演化过程对于不同的可信性特征属性均具备相应的复杂性特征, 这种综合的复杂性特征决定了软件系统的可信性. 软件系统可信性模型应该能够体现软件系统演化行为和软件系统演化的终极状态. 本文认为软件系统可信性演化模型应至少蕴含以下 3 种形式:

自然演化 涉及到软件在实时运行过程中的登录退出、功能的老化失效、性能的衰退等. 自然演化主要考虑软件在无外界环境影响的情形下, 自身不断运行下的演化规律. 这是一种最简单最基本的演化形式.

人为演化 涉及到软件的测试、维护、版本更新、补丁程序和软件知识库的更新等. 人为演化主要是针对软件本身的某个或某几个特征属性存在的缺陷和不足等, 通过人为干预的方式进行调整. 这种调整的目的是为了不断 “优化” 该软件, 使其具备 “更高” 的可信性. 但是, 对于单一或部分属性的局部优化调整是否能够达到全局综合可信性指标的整体优化目标尚缺乏理论支持.

可生存性 软件系统可信性是在外界环境不断变化下演化的. 一个可信的软件必须具有在外界环境中抗 “干扰”(抗病毒等) 能力, 也即具有可生存性. 软件系统的可生存性是指受到网络攻击、出现故障以及系统规模扩展过速的冲击下、部分组件受损, 系统仍能继续完成任务的核心活动、并恢复某些受损功能的行为特征 [14].

软件的自然演化确定了一个动力系统 任意给定软件的一组初始状态特征属性值 (也即软件从开始所具备的可信性出发), 随着时间演化, 软件的可

信性演化形成特征属性随时间变化的一条轨道 [15], 轨道的特性确定了软件可信性特征属性的行为.

软件的人为演化构成了一个 "类动力系统" (dynamical system-like) 具体而言, 软件在自然条件下演化一段时间后, 在某个时刻 t_1, 外界加入人为的调整 (例如版本的更新和升级等), 使得软件可信性演化会产生一个调整 (发生一次 "阶跃"); 经调整后的软件系统继续演化, 在下一时刻 t_2, 又一次发生人为调整的 "阶跃". 如此下去, 在这样一种实际的软件综合演化的考虑之下, 软件系统可信性的演化不再是传统的动力系统, 而是具有一个个 "阶跃" 特性的所谓 "类动力系统".

软件的生存演化问题对应动力系统的 "分支理论" 如果把软件系统受到的网络攻击、发生事故、产生故障等因素表示为外部条件的控制参量, 那么软件所确定的动力系统的外部条件通过这些控制参量的变化而改变. 相应于软件系统可信性及可生存性问题, 整体演化状态会在控制参量的某些分支值 (bifurcation values) 处发生结构性变化.

综上所述, 基于对软件可信性及其演化规律的科学认识, 将动力系统的基本思想、理论和方法 (特别是其统计分析方法和分支理论 [16,17]) 应用于软件可信性的研究, 对于从理论上刻画软件可信性及其演化规律和认识软件可信复杂性具有基本的重要性. 上述研究方法可以为建立软件可信性属性的度量和评估体系理论提供坚实的数学基础.

2 可信性演化动力学模型构造及分析

根据上节的分析, 软件可信性演化的数学模型可以构造为: $F : M \times N \to N$, 其中 M 为软件的特征属性空间, $M \subset \mathbb{R}^m$, m 是所要考虑的软件特征属性的个数; N 为参数空间, 用以刻画外界因素, $N \subset \mathbb{R}^n$, n 是参数的个数. 给定一组外界影响参数值, 就确定了软件所处的一个特定的外界环境. 综合考虑内部和外部演化的特性, 软件可信性的演化规律为 $F = f \circ g$. 这里演化规律 F 采取了复合映射的形式: f 体现了内部 (自然) 演化规律, 而 g 则体现了外部 (人为、可生存性) 演化规律. 从而软件可信性演化体现为动力系统 F 自身的迭代行为. 通过研究软件可信性演化 (即 $F^1, F^2, \ldots, F^k, \ldots$) 趋于极限状态时的行为, 可以定性和定量地分析软件可信性.

软件可信性的行为可能随着参数的变化而变化, 当行为发生变化时, 对应于系统的分支现象. 在分支现象发生时, 一般而言, 软件系统的可信性行为会发生突变. 为了进一步制定优化策略, 提高软件可信性, 我们必须研究在分支点处发生了何种类型的分支, 这是因为不同类型的分支代表软件可信性行为突变的不同特点.

以下, 通过 2 个实例对上述可信性分析方法进行说明.

例 1 考虑一个受外部因素影响且外部因素之间相互制约的例子: 软件系统运行中内存消耗量 (假设内存总量为全概事件) 受病毒影响的变化问题. 记 $x_n(\geqslant 0)$ 为时刻 n 时系统内存消耗量, 假设不同时刻感染病毒所消耗的相对内存量 (即 x_{n+1}/x_n) 相同, 用参数 $a(\geqslant 0)$ 表示; 并假设病毒两两相互竞争内存, 致使所抵消的内存量正比于 x_n^2, 用参数 b $(\geqslant 0)$ 表示, 则关于系统内存消耗量有

$$x_{n+1} = x_n(a - bx_n).$$

令 $y_n = bx_n/a$, 得到

$$y_{n+1} = ay_n(1 - y_n). \tag{1}$$

(1) 式即为生物模型中著名的 Logistic 映射 (见文献 [18]).

设 y_0 为 [0,1] 内的任意初值, 对于不同的参数 a 值, 系统长期演化的极限状态表现出不同的特征.

根据 x_n 与 y_n 的关系, x_n 的演化分支情况与 y_n 相同, 即随着 a 的值逐渐增加, $x_n(n \to \infty)$ 逐渐出现 2 倍分支, 4 倍分支, ..., 2^k 倍分支, ..., 当参数 a 超过 3.569946 时, 出现混沌现象 [19]. 由图 1 可知, 当 a 在 (0, 3) 中取值时, $x_n(n \to \infty)$ 仅收敛到一个值, 软件的内存演化趋势是可以预期的, 即软件是可信的. 当 a 在 $(3, 1 + \sqrt{6})$ 中取值时, 内存消耗量 $x_n(n \to \infty)$ 在两个固定值之间波动, 意味着内存受病毒影响开始处于不稳定状态, 即, 软件可信性演化复杂性开始增高. 其他分支情况可同理分析. 总之, 随着 a 的逐渐增加, 系统的分支现象越来越复杂 (见图 1), 导致系统的波动性逐步加强, 可信性演化复杂性逐渐增高, 直至最后进入混沌区 ($a \geqslant 3.569946$). 此时内存消耗量 x_n 的判定处于失控状态, 软件 "完全不可信".

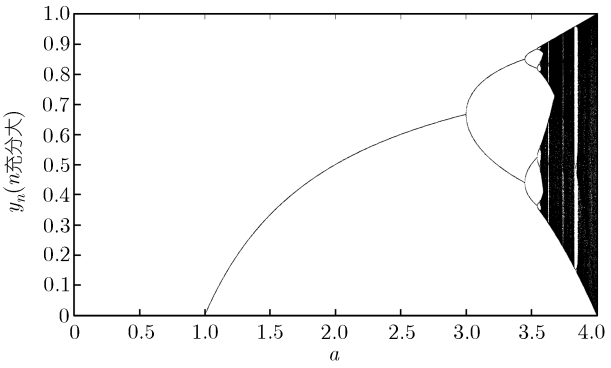

图 1 分支图 (0 <a<4) 纵轴表示系统长期演化后的内存消耗量取值状况, 横轴表示参数 a

例 1 说明了只有一个因素影响的情况, 下面考虑受内外两个因素作用的例子.

例 2 考虑一个受外部因素 (病毒攻击, 同例 1) 和内部因素 (例如考虑用户并发数, 假设最大用户并发数为全概事件) 综合影响的二维例子. 记 $x_n(\geqslant 0)$ 为 n 时刻的系统内存消耗量, 假设不同时刻新消耗的相对内存量相同, 用参数 a $(\geqslant 0)$ 表示; 由于病毒两两相互竞争内存, 致使所抵消的内存量正比于 x_n^2, 用参数 b $(\geqslant 0)$ 表示; y_n $(\geqslant 0)$ 为 n 时刻用户并发数, 而用户并发数必造成内存消耗, 记参数 c $(\geqslant 0)$ 为用户并发数与内存量的消耗调节参数; 假设每一时刻的用户并发数与上一时刻的内存消耗比例固定, 记为 d $(\geqslant 0)$, 则该系统可描述为

$$\begin{cases} x_{n+1} = x_n(a - bx_n) + cy_n, \\ y_{n+1} = dx_n. \end{cases}$$

令 $u_n = \frac{4b(x_n - \frac{a}{2b})}{a^2+2acd-2a}$, $v_n = \frac{4bc(y_n - \frac{ad}{2b})}{a^2+2acd-2a}$, 它可归结为天文学中著名的二维例子 ——Hénon 映射 [18,19]:

$$\begin{cases} u_{n+1} = 1 - \alpha u_n^2 + v_n, \\ v_{n+1} = \beta u_n. \end{cases} \tag{2}$$

其中 $\alpha = \frac{a^2+2acd-2a}{4}$, $\beta = dc$.

以 $\beta = 0.3$ 为例, 根据实际情况, 可以只考虑 $0.1225 < \alpha < 1.55$.

当 $0.1225 < \alpha < 0.3675$ 时, 系统可信性演化趋于一个确定状态, 其内存消耗量与用户并发数等特征属性都是可以预期的, 从而软件的可信性也是可以预期评测的.

当 $0.3675 < \alpha < 1.06$ 时, 系统可信性演化不再趋于一个确定的状态, 而是在若干有限状态间不断波动; 随着参数 α 的增加, 演化过程在内存消耗量和用户并发数的共同作用下呈现出越来越强的不确定性, 软件可信性演化的不确定程度随之增加, 即软件可信性演化复杂性逐渐增高. 尽管如此, 但是此时系统演化仍具有一定的规律性 (极限状态具有周期性), 因此我们对软件可信性仍可进行控制与预期.

当 $1.06 < \alpha < 1.55$ 时, 软件的内存消耗量和用户并发数两个特征属性的演化完全失去了规律性, 系统进入混沌状态 (如图 2 所示), 这意味着可信性演化复杂性的混沌现象导致软件 "完全不可信": 软件的各个特征属性混乱无序的进行演化, 甚至遍历于某些取值范围, 完全无法控制和预期. 系统演化的这种极度不稳定性, 表明了相应的软件是不可信的.

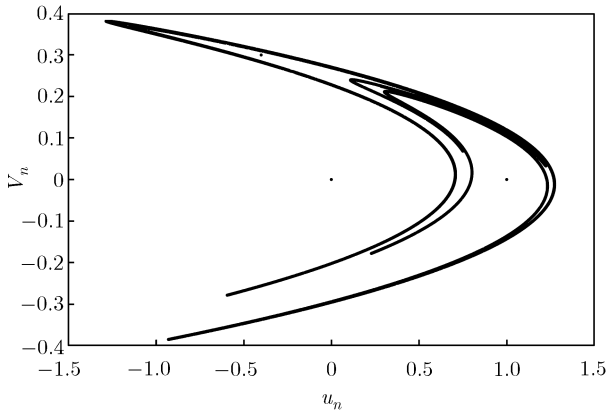

图 2 在参数 α=1.4, β=0.3 时的混沌图

上述两个实例说明了软件系统可信性演化与动力系统特征的对应关系, 诠释了软件可信性的动力学特征及其演化复杂性: 软件系统的行为及其结果可预期对应着(类)动力系统的一类动态行为; 行为状态可监测对应着(类)动力系统的轨道分析; 行为结果可评估对应着(类)动力系统的极限集行为分析; 行为异常可控对应着(类)动力系统的结构稳定和分支研究.

在另文中 [22], 我们将利用动力学统计分析方法, 定量地给出软件可信复杂性的统计度量模型, 以及软件系统不可信的动力学判据.

3 结论

在软件可信性研究领域, 可信计算的技术实践已经取得长足的发展, 在若干技术层面取得了非常

有意义的成果. 本文在可信计算技术实践和工作成果基础上, 对软件可信性基本科学问题进行探讨和研究. 本文结合动力系统的基本思想, 研究在各种内部和外部因素作用下软件可信性演化规律, 导出可信性演化的动力学机制.

本文通过对两个实例的建模分析, 说明软件系统的可信性是软件系统在动态开放环境下其行为的统计特性, 诠释了软件可信性的动力学特征及其演化复杂性: 软件系统的行为及其结果可预期对应着 (类) 动力系统的一类动态行为; 行为状态可监测对应着 (类) 动力系统的轨道分析; 行为结果可评估对应着 (类) 动力系统的极限集行为分析; 行为异常可控对应着 (类) 动力系统的结构稳定和分支研究.

参考文献

1 Open Trusted Computing (OpenTC) Consortium. General activities of OpenTC. [EB/OL]. Available: http://www.opentc.net/activities/[2006-3-1]

2 Reith M, Niu J, Winsborough W H. Engineering trusted management into software models. In: Proceedings of the International Workshop on Modeling in Software Engineering. Washington: IEEE CS Press, 2007. 9–15

3 Littlewood B, Wright D. The use of multi-legged arguments to increase confidence in safety claims for software-based systems: a study based on a BBN of an idealised example. IEEE Trans Softw Eng, 2007, 33(5): 347–365

4 Musa J D, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction, Application, Professional Edition: Software Engineering Series. New York: McGraw-Hill Press, 1990

5 Littlewood B, Strigani L. Software reliability and dependability: a roadmap. In: Proceeedings of the Conference on the Future of Software Engineering. New York: ACM Press, 2000. 175–188

6 Castro M, Rodrigues R, Liskov B. BASE: using abstraction to improve fault tolerance. ACM Trans Comput Syst, 2003, 21(3): 236–269

7 Triantafyllos G, Vassiliadis S, Kobrosly W. On the prediction of computer implementation faults via static error prediction models. J Syst Softw, 1995, 28(2): 129–142

8 Gashi I, Popov P, Strigini S. Fault tolerance via diversity for Off-The-Shelf products: a study with SQL database servers. IEEE Trans Depend Sec Comput, 2007, 4(4): 280–294

9 Banerjee S, Mattmann C A, Medvidovic N, et al. Leveraging architectural models to inject trust into software systems. In: Proceedings of the Workshop on Software Engineering for Secure Systems-Building Trustworthy Applications. New York: ACM Press, 2005. 1–7

10 Kurzyniec D, Sunderam V, Migliardi M. On the viability of component frameworks for high performance distributed computing: a case study. In: Proceeding of the 11th IEEE International Symposium on High Performance Distributed Computing. Edinburgh: IEEE Press, 2002. 275–283

11 Hasselbring W, Ruessner R. Toward Trustworthy Software Systems. Comput, 2006, 39(4): 91–92

12 Kirovski D, Drinic M, Potkonjak M. Enabling trusted software integrity. Oper Syst Rev, 2003, 36(5): 108–120

13 李未. 网络环境下海量信息组织与处理的理论与方法研究. 国家 973 项目任务书, G199903270. 1999

14 李未. 海量信息的协同性与可生存性的理论与实践. 国家 973 项目任务书, 2005CB321900. 2005

15 Smale S. Differentiable Dynamical Systems. Bull Amer Math Soc, 1967, 73(6): 747–817

16 Friedman A, Reitich F. Symmetry-breaking bifurcation of analytic solutions to free boundary problems: an application to a Model of Tumor Growth. Trans Amer Math Soc, 2000, 353(4): 1587–1634

17 Friedman A, Hu B. Stability and instability of Liapunov-Schmidt and Hopf bifurcation for a free boundary problem arising in a tumor model. Trans Amer Math Soc, 2008, 360: 5291–5342

18 Benedicks M, Carleson L. On Iterations of 1-ax2 on (−1,1). Ann of Math, 1985, 122(1): 1–25

19 Collet P, Eckmann J P. Iterated Maps on the Interval as Dynamical System. Boston: Birkhäuser Press, 1980

20 Lorenz E N. Deterministic Nonperiodic Flow. J Atmos Sci, 1963, 20: 130–141

21 Hénon M. A two-dimensional mapping with a strange attractor. Comm Math Phys, 1976, 50(1): 69–77

22 Zheng Z M, Ma S L, Li W, et al. Complexity of software trustworthiness and its dynamical statistical analysis methods. Sci China Ser F-Inf Sci, 2009, 52(9): 1661–1667