软件系统故障传播模型研究进展

王 珣 王轶辰
(北京航空航天大学可靠性与系统工程学院 北京 100191)
(可靠性与环境工程技术重点实验室 北京 100191)

摘 要 复杂软件系统的复杂性和不确定性引起了软件行为、交互行为以及故障行为的复杂性。在对复杂软件系统的可靠性、安全性等方面研究的过程中,故障的传播行为逐渐引起了学者的广泛关注,成为国内外学者的研究热点。回顾了故障传播的研究现状,对故障传播问题研究方向进行了梳理,重点对影响故障传播过程的两方面进行了详细的介绍,包括体系结构特征和故障类型。最后提出故障传播研究领域存在的挑战和未来的研究方向。

关键词 软件系统,软件体系结构,软件故障,故障传播,软件可靠性工程
中图法分类号 TP3-05 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.6.001

Research Progress on Error Propagation Model in Software System

WANG Xun WANG Yi-chen
(School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China)
(Key Laboratory of Reliability and Environmental Engineering Technology, Beijing 100191, China)

Abstract The complexity and uncertainty of complex software system bring about the complexity of software behavior, interaction, and error behavior. In the research process of complex software system reliability, security and other fields, error propagation has been a research focus, which attracts wide attention of domestic and overseas scholars. Firstly, the latest development of error propagation was summarized. Then the research direction of error propagation was presented. What's more, this paper compared and analyzed the factors affecting error propagation, including system architectural characteristics and fault type. Finally, the challenges and future research of error propagation were proposed.

Keywords Software systems, Software architecture, Software fault, Error propagation, Software reliability engineering

1 引言

随着软件系统在社会各个领域的作用的日益凸显,其复杂性急剧提升,复杂软件系统一旦失效将引起巨大损失。一个非关键组件失效是如何引起其他(关键)组件失效甚至引起系统失效的[1],复杂软件系统中的故障如何传播并引发系统失效,成为了目前的研究热点。

由大量局部自治软件系统持续集成、相互耦合关联而成的大型软件系统与其所作用的社会系统和物理系统密切相关,系统要素之间的耦合交互关系动态变化且日趋复杂,整个系统的行为难以通过各自治软件系统特征的简单叠加来刻画,我们称之为复杂软件系统[2]。

复杂软件系统表现出了迥异于传统软件系统的性质。首先,复杂软件系统规模庞大、结构复杂,使得其具有持续变化的特点,表现为软件无法事先确定并精确描述,复杂软件系统需要在运行过程中不断演进[3],软件固有的复杂性与开发过程中版本的修改都是导致软件错误的主要因素。其次,复杂软件系统中各个局部自治系统的开发过程、体系结构、软硬件平台等往往是异构的。该特性导致了组件之间相互关系的多样性和差异性,使得故障在异质成员之间传播时会引起不同强度以及不同类型的故障,即复杂软件系统内的故障行为复杂多样。此外,在复杂软件系统中存在着非线性相互作用、自组织和整体行为涌现现象[4],使得软件的行为更加复杂和难以预测,另一方面也导致复杂软件系统故障行为以及失效的原因具有明显的不确定性。此外,复杂软件系统还具有长生存、高可靠和使命关键等特征。复杂软件系统在承担关键使命的时期,不仅要求自身没有缺陷,而且要具有容错能力,以正确应对各种意想不到的异常,这就要求软件在有故障或错误存在的条件下仍能良好地运行。了解软件在异常情况下的特性至关重要,特别要了解故障是如何在软件中传播并影响软件的执行的。

复杂软件系统的特性对软件失效机理的研究提出了新的挑战,研究故障行为特点是分析复杂软件系统不确定性及可靠性等的基本问题。在电网、因特网和交通网络等领域中,关于级联故障(Cascading Failure)的研究有很多,是复杂网络研究的一个分支[5-7],多用于交通网堵塞分析以及电网停电影响

到稿日期:2015-05-18 返修日期:2015-09-06 本文受复杂软件密集型系统交互故障建模及可靠性仿真分析评价(9140C930301140C93)资助。
王 珣(1990-),女,硕士生,主要研究方向为软件可靠性工程、故障传播建模,E-mail:buaa_wangxun@163.com;王轶辰(1977-),男,博士,副教授,主要研究方向为软件可靠性工程、复杂软件系统测试技术。

分析研究[8]。如果一个或者少数几个函数发生故障，该故障可能会随着调用和依存关系传播至其它函数进而引发其它函数无法正常运行，最终导致部分或者整个系统崩溃，称之为“级联故障”[9]。对于级联故障的研究本质上就是研究故障在网络中的传播问题[10,11]。例如，吉林大学的学者们[12]在复杂网络的基础上，通过引入函数容错能力和软件故障强度建立复杂软件的级联故障模型，研究故障强度、初始故障节点和容错能力等因素对故障传播的影响[13]。

故障传播是软件失效机理的研究热点之一，对故障传播进行深入研究十分必要。众多相关文献表明，在深入了解复杂软件系统时，许多研究都涉及到故障传播问题[14-17]，如故障定位、故障检测、软件质量度量、可靠性预测等。研究者从自身的研究目的和应用方向出发，将故障传播作为后续研究的基础内容进行阐述，因而在系统类型、对软件的抽象、故障类型选取等方面都存在较大差异，所以对故障传播的认识和研究方法也不相同。研究者选取的系统类型包括嵌入式系统、分布式系统和 COTS 系统等[18-20]，将软件抽象成为模块化软件、基于组件的软件等[20,37,39]，有的研究者并不区分故障类型[23,24,26]，而有的研究者则重点研究一种类型的故障行为[65,70,74]，如数据错误、硬件错误等。由此可见，目前故障传播的研究现状仍旧比较零散，不成体系，并没有作为一个独立分支得到深入的剖析和研究。

从系统科学的角度看，软件故障的传播是在系统实体间发生的一个复杂的传播过程，这些实体包括物理实体、运行于单个或多个 CPU 上的进程、数据库中的数据对象、程序内的函数或程序中的语句等[21]。对传播过程进行深入研究，有助于理解软件故障的机理，认识内在规律。越来越多的研究者认识到故障传播对复杂软件系统的各分支的研究都有一定的影响[27,40,41,47]。而故障以何种机理传播，研究者对此存在两种观点，一种观点认为系统的体系结构决定了故障传播行为[23,25,29]，即同样的故障在不同体系结构的软件系统中会演化为不同类型或不同严重级别的系统失效。这种观点将故障传播的研究建立在系统结构分析的基础上，因此这些学者将故障传播的研究重点集中在故障在体系结构内传播的规律上。另一种观点认为不同类型的软件故障具有不同的特征，继而会演化为不同类型的系统失效，即不同的故障在同样的软件内其故障传播行为规律不同，故障的类型决定了传播的行为[68,69,73]，因此这些学者将故障传播重点集中在对故障特性的研究上。笔者总结了众多相关文献，将故障传播研究方向分类总结如图 1 所示。

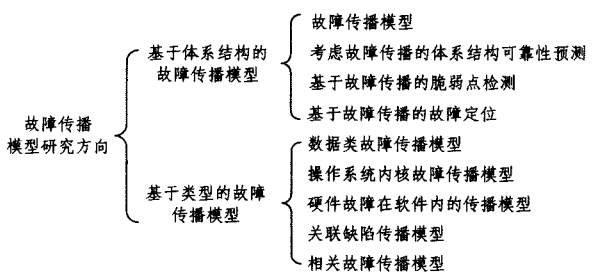

图 1　故障传播模型研究方向分类

在复杂软件系统中，由于内部或者外部原因，使得某个实体发生故障，由此可能引起其他实体发生故障的过程，本文称之为故障传播。根据现有的研究成果，故障传播过程由体系结构和故障类型决定。

本文主要回顾并总结了复杂软件故障传播的研究现状；第 2 节介绍基于体系结构的故障传播模型研究；第 3 节对典型故障类型的传播行为研究进行总结和比较；最后总结故障传播研究中目前存在的挑战和未来的研究方向。

2　基于体系结构的故障传播模型

从复杂软件系统自身的角度出发，故障传播行为由系统体系结构决定，即同样的故障在不同体系结构的软件系统中也会演化为不同类型或不同严重级别的系统失效，因此多数学者在体系结构模型的基础上提出刻画故障传播的特征量来描述传播行为。

2.1　故障传播模型

在软件工程领域，大多数经典的故障传播方法是基于故障注入或错误注入技术，结合进一步的数学分析与估计[21]。研究者多使用概率模型来描述对故障传播的认识，这些模型适用于定量的分析，对相关参数的定义和构造，体现了研究者对体系结构决定故障传播行为因素的认识[22]。

Abdelmoez[23-25]在 COTS 系统设计层面进行故障传播分析，使得获取系统开发早期阶段的信息成为可能。但是，需要十分细致和精确的 UML 描述才能获得精确可信的结果。该模型使用系统状态和报文信息来计算组件之间的故障传播概率，作者将错误传播概率作为体系结构的属性之一进行了研究，它反映了软件在运行时存在于组件 A 内的错误传播到组件 B 的概率：

$$EP(A,B)=Prob([B](x)\neq[B](x')\mid x\neq x') \tag{1}$$

在这个定义中组件的具体形式并不重要，组件可建模为函数，$[B]$代表了组件 B 的函数，它包含了 B 执行后的所有输出状态和结果；连接件 X 被认为是一个集合，集合中包含了 A 有可能传送到 B 的值 x。故障传播概率描述了组件 A 以概率 1 调用组件 B 的条件下错误传播的概率，为使模型更加接近实际使用时的情况，提出了无条件错误传播概率，表示为：$E(A,B)=EP(A,B)\times T(A,B)$。它由错误传播概率 $EP(A,B)$ 和概率转移矩阵 $T(A,B)$ 决定。概率转移矩阵的元素代表了连接件被激活的概率，即组件 A 调用组件 B 的概率。

基于这一理论，Hiller 等[26-29]引入了错误渗透率的概念，即输入信号中的一个错误渗透到一个输出信号中的概率，每一对输入、输出信号都有一个错误渗透率，表达式如下：

$$0\leqslant P^{M}_{i,k}=\Pr\{err\ in\ o/p\ k\mid err\ in\ o/p\ i\}\leqslant 1 \tag{2}$$

该定义的优点在于它独立于输入出现的概率。值得注意的是，当错误渗透率为 0 时，并不代表传入的错误没有造成任何影响，而是在组件的内部状态中导致了某种潜在错误，而由于某种原因并没有在输出显示出来。M. Hiller 以错误渗透率的提出为基础，相继提出了相对渗透率以及无权重的相对渗透率。错误渗透率针对一个输入输出对，而相对渗透率则是针对一个模块错误从输入传播到输出的概率，即：

$$0\leqslant P^{M}=\left(\frac{1}{m}\cdot\frac{1}{n}\right)\sum_{i}\sum_{k}P^{M}_{i,k}\leqslant 1 \tag{3}$$

相对渗透率未必能够反映一个模块整体的错误渗透率，但可作为一个抽象的度量值来表征模块之间相对的错误渗透能力。因此提出了无权重的相对渗透率，即：

$$0\leqslant P^{M}=\sum_{i}\sum_{k}P^{M}_{i,k}\leqslant m\cdot n \tag{4}$$

M. Hiller 认为多输入多输出的模块作为系统的中心模块，接收错误输入的可能性更大，将软件模块化并建立了故障

传播模型。该模型用于嵌入式系统模块化软件以及可信性系统的设计。

为便于模型的建立，研究者对组件进行了抽象和假设，虽然两个模型中参数都定义为条件概率，但实际上从组件模型中可以看出两个模型存在根本差异。M. Hiller 模型比 Abdelmoez 的方法更适用于实际应用，在后续的软件工具 PROPANE 的实现上证明了这一点。由于模型仅建立了软件部分的模型，这一理论只适用于复杂系统的软件部分。

文献[26]认为分布式软件是由错误控制模块(ECM)组成的，错误传播过程可以分为 3 步：首先，错误产生于源 ECM(ECM_S)；然后，错误从 ECM_S 传播出去；最后，导致目标 ECM(ECM_T)产生错误。借鉴 M. Hiller 提出的信号错误渗透率的概念，作者进行了进一步延伸，提出了错误传送概率和错误透明度的概念。错误传送概率指在 ECM_S 的输入发生了一个瞬时错误，并通过 ECM_S 的输出 M_j 传播至 ECM_T 的输入集，定义如下：

$$P_j^1=(Pr\{I\}/N)\cdot\sum_{k=1}^{N}Pr\{M_j\mid I_k\} \tag{5}$$

$Pr\{M_j\mid I_k\}$代表位于第 k 个输入 I_k 的错误通过 M_j 传播出去的概率。$Pr\{I\}$代表 ECM 输入集 I 发生错误的概率，在注入实验中 $Pr\{I\}=1$，N 代表模块 ECM 的输入个数。错误传送概率描述了错误传播的前两步，即 ECM_S 产生错误，并传播出去；而错误透明度描述了接收了错误输入 M_j 后 ECM_T 发生错误的概率，该参数描述了错误传播的最后一步，基于前两步的结果使 ECM_T 产生了错误。错误从 ECM_S 的输入传播至 ECM_T 的输出的过程如图 2 所示。

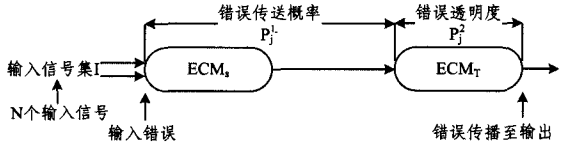

图 2　错误从 ECM_S 的输入传播至 ECM_T 的输出的过程

容错性作为影响故障传播的另一个重要因素，在前两个模型中并没有得到体现。故障在传播过程中，可能由于软件中设计了容错模块，使得故障被掩盖，并没有继续传播。因此，王健[12]在复杂网络的基础上提出了故障强度和容错能力两个参数，定义了故障强度 λ 的 8 个等级，等级越高则该故障影响其他函数的可能性越大。软件网络中每个节点(函数)对故障的处理能力称为节点容错能力 ρ，参考故障强度，容错能力也被划分为 8 个等级，等级越高表示节点容错能力越强，越不易受故障影响，容错能力与它拥有的调用关系数目(度)和调用关系紧密程度之和(强度)成正比例关系。此外还设置了函数调用频度作为每条边的权重，并以之作为元素构成了邻接矩阵。节点的故障感染率 P_i 由故障强度与容错能力的比值和函数之间调用的密切程度决定，表达式如下：

$$P_i=W_{ij}\times\beta_{ij};W_{ij}=w_{ij}/w_{max};\beta_{ij}=\begin{cases}1, & \lambda_i\geqslant\rho_i\\ \lambda_i/\rho_i, & \lambda_i<\rho_i\end{cases} \tag{6}$$

w_{max}为图 G 的最大权重，即最大的调用次数；W_{ij} 表示函数 i 与函数 j 的调用关系在软件网络中发生的概率，即调用关系的紧密程度。

前面介绍的模型均使用概率的方法对故障传播的可能性进行了计算，但是难以获得故障传播的具体路径。李果[30]使用蚁群算法求解扩散能力最强的故障传播通路，在小世界网络的基础上定义了故障扩散强度来表示故障通过某一条边进行扩散的能力，故障传播强度越大表示故障通过此边时越容易进行扩散，波及的范围也就越大。计算公式如下：

$$I_{ij}^k=w_s[w_pP_{ij}^k+w_ad_j^k/\sum_{j\in F_k}d_j^k];i\in F_{k-1} \tag{7}$$

基于体系结构的故障传播模型的准确性和精确性受到体系结构信息的影响。这与传染病传播模型的发展过程相近，以经典的 SI、SIS 以及 SIR 模型为基础，传播的载体在于人的接触网络，同时考虑年龄结构、随机性、人口流动以及跨地区的空间异质性等方面对模型进行扩展[31]，使模型更加贴近实际。随着复杂网络的提出，基于网络理论的传染病动力学成为研究的热点[32-34]。Muhammad Shafique 等对故障掩盖和故障传播属性进行建模，Abdelmoez 在研究中提到故障被掩盖的情况，但是并没有对其进行深入研究，与之不同，Muhammad Shafique 认为故障传播参数是错误掩盖参数的一个产物，它依赖于结构参数、控制流图和数据流图以及基本模块的执行概率。从故障掩盖角度入手，通过实验、算法和数学推导获得故障传播概率估计值[35]。

此外，除了使用概率模型方法对故障传播进行研究外，Jianfang Zhang[36]还应用统计的方法将故障传播模型用 $y=f(X+Z)$来表示，f 代表系统函数或传播函数，X 是系统的输入向量，Z 是 X 向量的错误向量，y 是系统的输出。该方法使用数理统计的方法将 $f(X)$用泰勒公式展开，并推导计算或估计错误 Z 分散分布时 y 的均值和方差。Devesh Bhatt 应用了区间数学理论，提出了一种基于区间运算分析的数据流模型，用于量化描述故障传播[37]。

在体系结构层面对故障传播的研究主要应用于考虑故障传播的体系结构可靠性预测、基于故障传播的故障定位以及基于故障传播的脆弱点检测。

2.2　考虑故障传播的体系结构可靠性预测

基于故障传播概率，研究者侧重于关注将故障传播考虑在内的可靠性预测，考虑故障传播对可靠性的影响，使预测结果更加准确。

Avizienis A 描述了组件故障导致系统失效的过程，如图 3 所示，该过程描述如下：假如组件的一个内部活动的代码实现存在缺陷——内部错误。当这部分代码被执行后，错误就会导致组件的一个内部故障；该故障一旦到达组件的接口，就会导致组件失效。而如果该故障被其他的内部活动指令所掩盖，就不会出现组件失效。类似可得：基于组件系统内，一个组件故障，如果故障传播到系统的接口，则该组件故障会导致系统失效；而如果故障被其他组件的指令掩盖，则不会导致系统失效[38]。

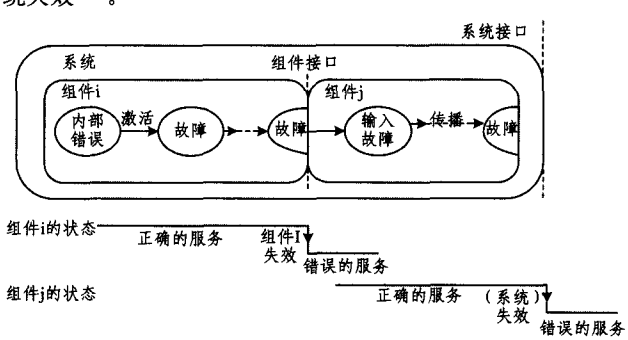

图 3　组件故障导致系统失效的过程

基于 Avizienis A 的认识，Cortellessa V 提出了考虑故障

传播影响的基于组件系统的可靠性分析模型方法。他认为基于组件的系统的可靠性可以定义为无系统失效发生的概率，影响因素有3个：1)每个组件的内部失效概率；2)每个组件的故障传播概率，即组件将接收到的错误输入传播至其输出接口的概率；3)传播路径概率，即每一条可能的从组件到系统输出的故障传播路径的概率。前两个因素是组件的固有属性，而第三个因素则依赖于系统的结构，且假设内部失效概率与故障传播概率相互独立。Mohamed 和 Zulkernine[39] 提出了故障传播分析的另一种方法并将其应用于可靠性评估，该模型可视为 Cortellessa[40,41] 模型的一个分支。

Popic P 建立了考虑故障传播的贝叶斯可靠性预测扩展模型，该模型是 Cortellessa 研究的扩展。他们认为故障传播概率是软件体系结构的一个重要的结构参数[42]，首先利用 Abdelmoez 的方法估计故障传播概率，并利用 UML 用例图、序列图以及部署图的相关信息建立所需的参数，包括某一系统行为发生的概率、组件 C_i 在场景 j 的失效概率、连接器的可靠性等；利用计算得到的概率以及故障传播概率计算基于组件的系统的可靠性。

Popic P 认为一个组件内的一个错误通常会导致系统失效，同时该错误也会传播到其他组件，从而影响其可靠性。Cortellessa V 则认为每个组件内部失效的概率以及错误传播的概率是彼此独立的。基于该独立假设，当一个组件失效时，该组件常常将错误传输到下一个组件，而与该组件是否收到之前组件的错误输入无关[41]。

但是也有学者提出了质疑，认为 Popic P 的立即失效假设与故障传播到其他组件的原因相冲突，由于错误的输出会被其他组件的指令重写，因此组件仍然能够产生正确的输出。而 Cortellessa V 仅考虑了会产生错误输出的失效，忽略了其他类型的失效。此外，学者一般假设软件为顺序执行模型，而无法对并行执行以及容错执行进行建模。因此 Thanh-Trung Pham 将故障传播分析和多种执行模型结合在一起，提出新的可靠性预测模型[43]。

Thanh-Trung Pham 为解决已有的考虑故障传播的可靠性预测模型只能对顺序执行模型进行预测，即在任何时刻只有一个组件正在执行的问题，将循环结构、平行结构转换为顺序结构或分支结构，在考虑失效概率和故障传播概率的基础上，求解转换后的模型的概率信息，从而实现在不同的控制流结构上充分考虑故障传播的不同影响，建立了马尔科夫可靠性预测模型；同时将故障传播概率定义为一个组件内的故障传播到其他组件的可能性[44]，同样支持 Avizienis A 的故障传播过程的描述。

Thanh-Trung Pham 在后续研究中，又加入了容错机制以及并发故障对可靠性预测模型的影响，认为主备份容错结构具有一个主要的功能组件以及一系列的备份组件软件。在运行时这些组件的执行顺序取决于故障检测和故障处理，因此该主备份容错结构将导致不同的故障传播路径[45]产生。此外，并发结构会在软件并发执行时使同一个故障经过不同的指令，进而产生不同的故障类型以及故障传播路径。

此外，Lance Fiondella 和 Swapna S. Gokhale 在故障传播的基础上也考虑了容错性[46]。基于体系结构的可靠性评估一般假设单点故障，即组件故障会引起系统失效，但是其忽视了容错机制等的影响，所以评估结果偏向悲观，由此引出对故障传播的研究。对软件中故障受容错机制的影响进行分析，目的是解决单点故障假说，提高可靠性评估精度。

2.3 基于故障传播的故障定位

近年来，在人工智能领域和软件工程自动化领域已经出现了很多定位技术，其中基于依赖关系的故障定位受到广大学者的关注，一般依赖关系被分为两类：控制依赖和数据依赖。通过计算可疑度进行故障定位，可疑度越高则故障在该基本块内的可能性越大。传统的故障定位方法未考虑故障传播的影响，在获得可疑度最高的模块后并不能确定故障究竟是产生于该模块，还是该模块仅仅传播了产生于前驱模块的错误，因可疑度计算的不准确而求得了较大的可疑度值。因此在考虑故障传播的故障定位研究中，研究者在可疑度的计算中考虑了故障传播，对可疑度的计算进行了修正。

马凯[47]在研究中提出了一种基于故障传播的故障定位方法，结合了基于执行覆盖以及基于依赖关系的两种故障定位方法，同时考虑故障传播对定位的影响。故障传播是通过程序中存在的依赖关系进行的，因此传播可疑度的计算是建立在计算依赖对可疑度 $\theta^{\Delta}(\varepsilon)$ 的基础上的。依赖对的传播率计算方法如下：

$$W(b_j,b_k)=\frac{\theta^{\Delta}(b_j,b_k)}{\sum\limits_{\forall(*,b_k)}\theta^{\Delta}(b^*,b_k)} \tag{8}$$

其中，分子部分代表依赖对 b_j 与 b_k 的可疑度，b_k 为 b_j 的后继块之一，程序的故障状态可能会通过依赖对 (b_j,b_k) 由 b_j 传播到 b_k；分母部分则代表所有以 b_k 为后继块的依赖度之和，二者的比值作为指标衡量 b_j 对 b_k 的传播率。

马凯认为错误会通过控制依赖边传递到后续模块，而王煜[48]的方法则认为错误是通过数据依赖边在语句之间传播的。程序依赖图[49,50]是一个有向图，节点表示程序的状态，边用来表示控制依赖和数据依赖。控制依赖边上的权值表示分支条件的判断情况，数据依赖边上的权值表示沿着该边传递的变量。在程序依赖图的基础上通过转换得到联合依赖图。他在此基础上给出了可疑度的计算方法：

$$final\_susp(n_i)=suspiciousness(n_i)+\sum_{\forall edg(n_i,n_j)\in edg(n_i,*)}[k_{edg(n_1,n_j)}*suspiciousness(n_j)] \tag{9}$$

由此可以看出语句 n_j 的错误有两个来源，一个是其本身产生的，另一个是从前驱节点传播过来的。式(9)中括号的部分表示语句 n_i 来自父节点 n_j 的错误。k 代表每条数据依赖边传播错误的概率[51]。

为了准确地进行故障定位，何加浪[52]首先找到可疑度最大的节点 V_{max}，接着分析与其直接前驱节点是否存在故障传播趋势，如果存在，则递归处理该前驱节点直到到达某个不再存在这种传播趋势的节点 V_o，然后以该节点为起点、以节点 V_{max} 为终点建立一条虚边来修正 V_o 的可疑度。由此，考虑故障传播因素的影响后可以准确区分以下两种情况：1)故障位置在 V_{max} 前驱节点中，只是经由边 (V_o,V_{max}) 到 V_{max} 处；2)故障位置就是在 V_{max} 处，使得故障定位更加准确。

2.4 基于故障传播的脆弱点检测

软件脆弱点可以理解为在软件系统中存在着某些固有的弱点或缺陷，是软件代码内部自身的缺陷。它们的执行能显

式或隐式地对系统造成破坏。此外，软件系统中的各个部分或单元对环境扰动的抵御能力各不相同，对环境扰动最敏感的部分也称为软件脆弱点[53]。

Jeffrey Voas 使用故障传播分析仿真系统和组件的行为，首先通过故障传播仿真，确定不能接受的输出集合，这既包括系统 S 也包括对组件 C_i 而言不可接受的行为，接着使用静态切片算法设置故障恢复点。为了更加具有成本效应，应尽量减少错误恢复机制的数量。仿照多米诺骨牌坍塌的原理，寻找最少的位置设置障碍点，从而阻止大量的多米诺骨牌坍塌[54]。

M. Hiller[27]则在故障传播分析过程中确定故障检测和恢复机制的安放位置，将软件模块化，一个模块被认为是一个具有多个输入与输出的黑盒，模块之间使用不同形式的信号来进行通信。信号中的错误可沿着许多不同的路径传播至系统输出，基于信号错误渗透率，构造了信号的传播树算法，分析某一信号对系统输出的影响。根据该算法，系统输入到系统输出之间，具有最高权重的路径就是错误最有可能传播的路径。接着在最有可能传播错误的路径上寻找脆弱模块，引入错误曝光率的概念，它反映了一个模块遭受错误的数量，定义为组件 M 的所有传入边的权重和与传入边数量 N 之比：

$$X^M=\frac{1}{N}\sum \text{weight of all incoming arcs of } M \qquad (10)$$

将 EDM 放置在错误曝光值高的地方，更加具有成本效应，将 ERM 放置在错误渗透率高的地方，更加具有成本效应。

Arshad Jhumka 在提出错误转移概率和透明度的概念后，提出了两个衡量 ECM 脆弱性的指标，分别是影响值和隔离量。影响值代表了源 ECM 对目标 ECM 的影响，它的计算方法是转移概率 P_j^i 和透明度 P_j^2 的乘积。隔离量代表了在考虑其他所有 ECM 的情况下，源 ECM 不影响其他 ECM 的概率。两个值分别代表了源 ECM 和目标 ECM 之间直接和间接的交互。理论与相关实验分析表明，影响值高的 ECM 适合安放 EDM 和 ERM。

以上两种方法均基于故障传播来确定脆弱点，选择合适的故障检测和恢复机制位置。假设系统只受输入错误的影响，使用错误传播分析来确定故障检测机制的安放位置(PA)与传统的经验推断方法(EA)相比，从内存和执行时间要求来看，PA 方法比 EA 方法减少了约 40%；两种方法的检错覆盖率是一样的。因此错误传播的方法在获得同样的检错覆盖率的条件下，缩减了需要的内存和执行时间[55]。

研究者多集中于使用概率的方法对脆弱点检测进行研究；获得与故障传播和软件体系结构相关的参数，给出参数理论的计算方法；通过故障注入实验进行验证，将理论的参数值与实验获得的参数值进行对比和分析，从而证实理论研究的正确性。

故障的传播是通过系统中的交互作用进行的。在基于体系结构的故障传播模型中，研究者多认为传播是通过调用、控制流或数据流进行的，而实际上复杂软件系统内交互作用十分复杂，并不是单一的。

3 基于类型的故障传播模型

在同样的软件系统内，不同的故障传播具有不同的行为特点，从故障的角度出发，研究者细致和深入地研究了典型故障在具体类型系统中的故障传播的规律。实体之间传播的故障类型有许多种，比如物理实体之间的硬件故障、程序语句之间传播的软件故障以及函数之间错误的传递等[56]。

按照故障的来源，可以将故障类型分为硬件故障和软件故障；按照故障在系统中的传播以及对系统的影响，一般可以将故障导致失效的过程归结为两类模型，即立即失效模型和传播模型[57]。立即失效模型描述了故障发生后，除了发生故障的部件以外，系统中其他部分都不会受到该故障的影响。实际上，由于故障总会在系统中传播一定的距离[58]，完全的立即失效模型几乎是不存在的[59]。传播模型描述了故障发生后其结果会在系统中传播一定的范围[60-62]。本节总结了研究者在选取不同故障类型进行研究时，对故障传播认识的不同以及研究结论的差异；除介绍软件故障以及硬件故障的传播规律的研究外，还总结了相关故障的研究。相关故障不局限于从单个故障着眼，而是具有相关关系属性的故障。

3.1 数据类故障传播模型

M. Hiller 团队重点对数据错误的相关规律进行了研究，他们使用故障注入的方法，在模块的输入信号中注入故障，每一次只在一个输入信号注入一个故障，并记录该模块的输出，比较无故障的软件运行情况以及注入故障后的软件运行情况，追踪故障传播的路径，分析模块化软件中数据错误的传播[63]。与其前期的研究[27]不同，其进一步认为，存在于原输入信号中的错误可能会沿着不同的路径传播到系统的输出，从而影响系统输出的结果，因此模型的根节点不局限于系统的输入，而是将在分析中所有感兴趣的信号作为根节点。M. Hiller的研究目的是分析软件中的脆弱点，寻找系统中最容易传播错误的脆弱点，通过脆弱性分析可以识别出哪些组件的可信性结构和设计机制是最有效的。

与 M. Hiller 不同，文献[64]介绍了软件缺陷行为的一个实证研究：数据状态错误(Data-state Error)的传播。当缺陷被激活时就会产生数据状态错误，影响程序的数据状态。如果执行的输出也受到影响，那么该故障就会传播。研究结果表明数据状态错误在模拟故障代码时具有一个典型的性质：给定一个输入和指定的注入故障的位置，所有的状态错误或者都会传播并影响输出，或者全都不会。这一性质表明数据状态错误是有序的行为，软件行为也不是理论上描述的那样不可预测。此外，如果对于给定的输入和指定的位置，故障行为都相同，那么在进行模拟仿真时就很容易获得故障行为，不需要考虑仿真的故障是否能够代表实际的故障。

3.2 操作系统内核故障传播模型

操作系统是计算机系统的核心与基石，而内核是操作系统最基本的部分，为众多应用程序提供对计算机硬件的安全访问。内核对时间和产生于硬件的错误极为敏感，部分研究者重点研究了操作系统内核错误的传播规律。

A. Johansson 和 N. Suri[65]研究了内核和设备驱动程序之间的故障传播，并认为设备驱动程序是原子单元，这些原子单元只与操作系统内核进行交互，而不是彼此之间进行交互的，并在定义参数时只考虑驱动程序与内核之间的交互；讨论了故障是如何通过使用操作系统以及为操作系统提供服务而传播的。与之不同的是，T. Jarboui 与 J. Arlat[66]并不关注设

备驱动和内核模块之间的错误传播路径，只关注内核子系统之间的故障传播，在 Linux 内核中故障传播的渠道，提取了调用关系图，关注基本的内核子系统的调度、内存管理、同步、文件系统支持和基本的通信；描述了在不同的内核栈层次中进行故障注入时的不同故障模式：在较高层次接近应用层的错误更容易被检测到，表明错误处理多在设备驱动程序的入口完成，在驱动程序内的错误处理大部分不能被完整完成。

与 Jarboui T 和 Arlat J 的考虑类似，Drebes R J[67] 认为模块之间通过函数调用进行交互，这同样代表了模块提供可使用的服务。他们还认为错误可以通过这样的接口进行传播，但是更加关注的是直接的模块交互；研究了整体操作系统内核的内部模块错误，提出了对故障进行分组、隔离，从而阻断传播。抽象出核心模块之间的关系，识别故障在模块之间的传播路径，提出一项根据模块提供的功能对模块进行分组的技术，最后根据在单个隔离环境中的性能开销来评价该模块分组技术。

3.3 硬件故障在软件内的传播模型

复杂系统内软硬件交互也是产生软件故障传播的一个因素，因此研究者在选择故障类型时，硬件故障在软件内的传播也是研究热点之一。按照故障的持续时间，可以将硬件故障分为瞬时故障、永久故障以及间歇故障[68]。永久故障来源于硬件的制造缺陷，除非采取一定的修补措施，否则永久故障将一直存在。因此，研究者考虑的硬件故障类型主要包括缓存器瞬时错误和间歇性硬件错误等。

存储系统中发生的故障中有 98%都是瞬时故障[69]，且缓存器效率高的瞬时故障发生的概率大，因此学者研究了发生在缓冲存储器本身的或者是在处理器寄存器中或者是两者兼有的瞬时故障在软件中的传播规律。研究了产生于缓冲存储中本身或处理寄存器的故障在缓冲存储系统中的传播问题。设计此类错误恢复机制；使用离散时间马尔科夫模型预测错误传播的范围[70]。

间歇故障会在某段时间内连续发生，大多数间歇性硬件故障会导致系统崩溃。故障避免技术使用硬件方法避免间歇性故障，但还是会有类似的故障发生并传播到软件，因此研究者在软件层面上进行检测、诊断和恢复，削弱该类故障的影响。Rashid L 研究了间歇性硬件错误在软件内的传播规律。解答了哪一部分间歇性故障会导致程序崩溃；而可以导致程序崩溃的故障在程序崩溃之前是如何在程序内传播的。文献[71]描述了故障模型，并创建了一个崩溃模型来确定因间歇性故障而发生程序崩溃的位置[71]。

在实际测试过程中，很多软件故障并不是相互独立的，它们存在着某种相关关系。从故障自身方面来分析，这种相关关系的存在是由它们之间存在控制流和数据流所综合导致的。由于存在这种相关关系，若故障 A 一旦出现，可能会导致与之相关的故障 B 随之出现，从而对故障传播产生影响。从故障的角度对故障之间存在的相关关系进行研究，研究者认为故障之间存在某种相关关系，因故障彼此关联而影响故障在软件内传播，其中研究包括关联缺陷与相关故障。

表 1 总结了故障之间相关关系的研究。表中第 1 列为文献名称；第 2 列为文献的发表年份；第 3 列为文献的关键字，用于描述涉及的内容和技术。

表 1 故障之间相关关系的文献总结

文献名称	发表年份	关键字
Study of a Multi-component System with Failure Interaction	1985	可靠性、多组分的系统、故障交互
Failure correlation in software reliability models	2000	马尔可夫过程、相关分析法、软件可靠性
Software reliability growth models incorporating fault dependency with various debugging time lags	2004	软件可靠性、故障检测、程序排错
Modeling and Analysis of Correlated Software Failures of Multiple Types	2005	马尔科夫更新过程、软件可靠性
考虑故障相关的软件可靠性增长模型研究	2007	软件可靠性增长模型、故障相关性；、测试环境、运行环境
一种考虑缺陷关联的软件可靠性增长模型	2008	软件可靠性增长模型、非齐次泊松过程、软件缺陷关联
Path-Based fault correlations	2010	错误状态、故障相关、路径敏感
引入关联缺陷的回归测试技术研究	2010	关联缺陷、回归测试用例集缩减、HGS 算法
基于缺陷关联的静态分析优化	2014	静态分析、状态切片
一种故障相关多单元系统可靠性分析	2014	故障相关、表决系统、故障状态、马尔科夫过程

3.4 关联缺陷传播模型

研究者认为软件中的关联缺陷是一种比较普遍的现象，某些缺陷的存在与否可能导致其他缺陷检测率的变化。软件关联缺陷是造成软件失效关联的根源。刘新忠对缺陷的关联关系进行了描述、分析和研究[72]。如果缺陷 Da 的缺陷检测能力受到 Db 是否存在的影响，则称缺陷 Da 与 Db 为关联缺陷。关联缺陷可分为两类：正关联缺陷和负关联缺陷。

意识到关联缺陷的存在后，研究者重点研究如何检测、剔除和杜绝关联缺陷。景涛[73] 提出了一种缺陷放回的测试方法来剔除关联缺陷，即在一定度范围内随机放回已被检测到的软件，并以实验数据分析考虑关联缺陷的缺陷放回方法的能力和效率。Bishop 等人[74] 指出可以用失效屏蔽效应来解释软件失效之间的关联关系，并可以通过适当的软件设计来杜绝失效关联。

大多数的非齐次泊松过程（NHPP）可靠性增长模型都有这样的假设：每个缺陷的严重性和被检测到的可能性相同，在排除一个缺陷时不引入新的缺陷。在引入关联缺陷后，这样的假设被认为不符合实际的情况，Katerina 和 Trivedi[75] 认为实际测试过程中软件失效不是相互独立的，并提出了一种马尔科夫更新模型来对有失效关联的软件可靠性进行建模。张荣辉引入了考虑缺陷关联的软件可靠性增长模型[76]，他指出由于缺陷之间存在着关联关系，而且在排除缺陷时有可能引入新的缺陷，从而软件本身固有缺陷数会增加，最终影响软件可靠性增长模型的预测结果。在考虑软件缺陷关联关系的基础上对缺陷进行分类，提出一个改进的 NHPP 软件可靠性增长模型。

在测试过程中人们发现了软件缺陷之间的关联关系，因此研究者将关联缺陷的应用重点放在了软件测试领域，如优化静态分析、回归测试用例集优化以及何时停止测试等具体的问题上。缺陷关联是一种有效降低缺陷确认负担的静态分析优化技术[77]。张大林等[78] 提出一种可靠的基于缺陷关联的静态分析优化方法，该方法能够在静态缺陷检测过程中自

动发现可靠的缺陷关联关系，结合这些关联关系，能够对静态检测工具所报告的缺陷进行分组。孙金珊[79]提出了缺陷的5种关联，包括缺陷的依赖关联、重复关联、相关关联、文件关联和附件，主要用于刻画缺陷与缺陷之间、缺陷与其相关文件之间的关系。将关联缺陷引入到回归测试用例集的研究中，考虑关联缺陷对选择回归测试用例集的影响，优化回归测试用例集顺序，缩小回归测试用例集，使其方法或算法更完美、更有效。

3.5 相关故障传播模型

研究者对相关故障的研究集中于相关故障对可靠性的影响。Wu Kang 等[80]将相关的故障分为两类，研究了这两类故障对软件可靠性的影响。Dai Yuanshun 等[81]提出了故障相关性的3种形式，建立了考虑故障相关性的、基于马尔科夫更新过程的软件可靠性增长模型。Huang Chinyu 等[82]在建立模型时，将软件故障之间的相关性分为两类（一类是独立的故障，另一类是相关的故障），对这两类故障进行分别建模。但是，对故障相关性进行分类只是一种简单的非形式化的方式，实际情况下，故障之间的相关性是非常复杂的，且随着测试的进行，故障之间的相关关系是不断改变的。

因此，赵婧等[83]对故障相关性进行形式化描述，提出了一种随机过程类 NHPP 考虑故障相关性、测试环境和运行环境差别的模型。在有向图基础上，重点形式化分析了软件故障相关性，提出了相关故障出现率的概念，并且分析了随时间变化的相关故障出现率，最后建立了既考虑故障相关性又考虑测试剖面与运行剖面差别的软件可靠性增长模型。

随着研究的深入，对故障相关进行了定量的刻画，故障的发生对另一故障是否发生产生的影响体现在对故障率的影响上。Murthy 和 Nguyen[84]总结了两种故障相关的类型：类型I故障相关，即当一个部件发生故障时，它会以一定概率引发系统中其他部件发生故障；类型II故障相关，即当系统中一个单元发生故障时，它会影响到其他单元的故障率。在故障相关多单元系统可靠性分析[85]中，提出了由一个主用单元和多个次要单元组成的一类多单元系统，与主要单元存在第I类故障相关关系。通过系统分析，得出了主用单元故障率函数。通过单元故障数将系统状态分为故障状态和正常状态两类，建立马尔科夫状态转移矩阵和一个一阶线性微分方程组，通过求解方程得出系统瞬态可用度及可靠度。

由以上分析可以看出，故障传播模型研究可用于软件测试领域中优化静态分析、回归测试用例集优化以及何时停止测试等具体的问题。此外，传统可靠性模型多假设故障之间彼此独立，故障之间的相关关系研究也可应用于可靠性模型的改进。

4 挑战与趋势

尽管目前存在众多故障传播方面的研究成果，但是由于对软件体系结构刻画不准确等问题，软件系统的故障传播问题的研究仍然存在很多机遇与挑战。

（1）交互作用引起故障传播，即故障的传播是通过系统中的交互作用进行的。软件系统交互行为十分复杂，而目前的研究缺乏对交互作用的全面认识和分类，已有的故障传播模型很难准确描述各种交互作用，不能完整体现出交互作用给故障传播带来的影响，从而无法准确地刻画体系结构内的故障传播。

（2）在实际测试时可以发现软件故障的类型众多，而在不同类型故障传播的研究中，故障类型分类不全，已有研究的软件故障类型仅有数据错误以及操作系统内核错误两种类型，且研究者数量仍较少。因此将软件故障进行分类，并对不同类型的软件故障的传播规律进行研究仍有待加强。

（3）大多数研究认为传播是一个动态的过程，但是传播的错误是静态的，即错误在传播过程中不发生变化，这与实际情况不符。在今后的研究中，应考虑到错误在传播的过程中会发生变化，传播至不同的软件位置时错误的表现不同。错误在传播的过程中其特性会发生改变，甚至表现出与初始的错误完全不同的特性。这体现出不同类型的故障在复杂的交互过程中表现出了不同的失效效果。

（4）研究者对故障传播机理存在两种观点，一种观点认为故障传播由系统体系结构决定，另一种认为故障传播过程由故障类型决定，不同类型的软件故障具有不同的特征，继而会演化为不同类型的系统失效。因此存在体系结构内故障传播规律和典型故障类型传播两个研究方向，但是很少有研究者同时考虑体系结构和故障类型两方面因素对故障传播的影响。

结束语 作为软件故障机理研究领域的一个分支，故障传播是比传统软件由错误到失效机理研究更深层次的本质性研究，其研究价值也正得到越来越多人的认可和重视。从文中的介绍可以看出，软件系统故障传播的研究刚刚起步，已有的研究成果使我们对故障传播过程有了初步的认识，但是距离实际的软件系统故障传播机理和规律的揭示还有很大的距离。

参 考 文 献

[1] Sarshar S. Analysis Error Propagation between Software Processes in Source Code[D]. Norway: Østfold University College, 2007

[2] Wang Huai-min, Wu Wen-jun. The growth structure and adaptive evolution of complex software systems[J]. Science China Information Sciences, 2014, 44(6): 743-761(in Chinese)
王怀民，吴文峻. 复杂软件系统的成长性构造与适应性演化[J]. 中国科学：信息科学，2014，44(6)：743-761

[3] Liu Xiao-ping, Tang Yi-ming, Zheng Li-ping. Survey of Complex System and Complex System Simulationi [J]. Journal of System Simulation, 2008, 20(23): 6303-6315(in Chinese)
刘晓平，唐益明，郑利平. 复杂系统与复杂系统仿真研究综述[J]. 系统仿真学报，2008，20(23)：6303-6315

[4] Zhang Si-ying. Complexity Science, Rules of the Whole Systems and the Qualitative Researches[J]. Complex Systems and Complexity Science, 2005, 2(1): 71-83(in Chinese)
张嗣瀛. 复杂性科学，整体规律与定性研究[J]. 复杂系统与复杂性科学，2005，2(1)：71-83

[5] Albert R, Barabasiá A-L. Statistical mechanics of complex networks[J]. Review of Modern Physics, 2002, 74(1): 47-97

[6] Newman M E J. The structure and function of complex networks[J]. SIAM Review, 2003, 45(2): 167-256

[7] Wang Jian, Liu Yan-heng, Mei Fang, et al. Modeling Cascading Failures for Internet Based on Congestion Effects[J]. Journal of Computer Research and Development, 2010, 47(5): 772-779(in Chinese)
王健，刘衍珩，梅芳，等. 基于网络拥塞的 Internet 级联故障建模

[J]. 计算机研究与发展,2010,47(5):772-779

[8] Dobson I. Where is the edge for cascading failure: Challenges and opportunities for quantifying blackout risk[C]// Proc. IEEE Power Eng. Soc. Gen. Meeting. 2007:1-8

[9] Wang Jian, Liu Yan-heng. Modeling software faults propagation [J]. Epl, 2010, 92(6): 1637-1649

[10] Bao Z J, Cao Y J, Ding L J, et al. Dynamics of load entropy during cascading failure propagation in scale-free networks[J]. Phys. Lett. A, 2008, 372: 5778-5782

[11] Lehmann J, Bernasconi J. Stochastic Load-Redistribution Model for Cascading Failure Propagation[J]. Phys Rev E Stat Nonlin Soft Matter Phys, 2009, 81(3): 227-248

[12] Wang Jian, Liu Yan-heng, Liu Xue-lian. Model for Cascading Faults in Complex Software[J]. Chinese Journal of Computers, 2011, 34(6): 1137-1147(in Chinese)
王健,刘衍珩,刘雪莲. 复杂软件的级联故障建模[J]. 计算机学报,2011,34(6):1137-1147

[13] Liu Xue-lian. Software fault propagation analysis and simulation [D]. Changchun: Jilin University, 2011(in Chinese)
刘雪莲. 软件故障传播分析与仿真[D]. 长春:吉林大学,2011

[14] He Jia-lang, Meng Jin, Zhang Kun, et al. A Fault Propagation-aware Program Fault Location Method[J]. Journal of Electronics & Information Technology, 2011, 33(9): 2192-2198(in Chinese)
何加浪,孟锦,张琨,等. 一种故障传播感知的程序故障定位方法[J]. 电子与信息学报,2011,33(9):2192-2198

[15] Jhumka A, Leeke M. The Early Identification of Detector Locations in Dependable Software[C]// IEEE International Symposium on Software Reliability Engineering. IEEE Computer Society, Hiroshima, Japan, 2011: 40-49

[16] Pan Wei-feng, Li Bing. Software quality measurement based on error propagation analysis in software networks[J]. Journal of Central South University (Science and Technology), 2012, 43(11): 4339-4348(in Chinese)
潘伟丰,李兵. 基于软件网络错误传播分析的软件质量量度[J]. 中南大学学报(科学版),2012,43(11):4339-4348

[17] Gao Xiao-qing, Peng Tao. A Line Detection Algorithm Based on Error Propagation [C] // Chinese Control Conference. 2007: 493-496

[18] Voas J. Error propagation analysis for COTS systems[J]. Computing & Control Engineering Journal, 1997, 8(12): 269-272

[19] Kukla G. Quantitative Analysis of the Error Propagation Phenomenon in Distributed Information Systems [C] // Intelligent Information and Database System First Aasin Conference. 2009: 202-207

[20] Shin K G, Lin T H. Modeling and measurement of error propagation in a multimodule computing system[J]. IEEE Transactions on Computers, 1988, 37(9): 1053-1066

[21] Morozov A, Janschek K. Probabilistic error propagation model for mechatronic systems[J]. Mechatronics, 2014, 24: 1189-1202

[22] Sarshar S, Simensen J E, Winther R, et al. Analysis of Error Propagation Mechanisms between Software Processes [C] // Safety and Reliability Conference. Stavanger, Norway, 2007: 91-98

[23] Ammar H H, Nassar D, Abdelmoez W, et al. A framework for experimental error propagation analysis of software architecture specifications[C] // Proceedings of the Int. Conf. on Computer and Communication Engineering. Kuala Lumpur, Malaysia, 2006: 9-11

[24] Abdelmoez W M, Nassar D, Shereshevsky M, et al. Error Propagation In Software Architectures[C]// IEEE International Symposium on Software Metrics. IEEE Computer Society, 2004: 384-393

[25] Nassar D E, Rabie W A, Shereshevsky M, et al. Estimating Error Propagation Probabilities in Software Architectures [M]// Handbook of Graph Grammars & Computing by Graph Transformation. 2006

[26] Jhumka A, Hiller M, Suri N. Assessing inter-modular error propagation in distributed software[C]// Symp. on Reliable Distributed Systems Distributed Software. 2001: 152-161

[27] Hiller M, Jhumka A, Suri N. An approach for analysing the propagation of data errors in software[C]// Proceedings of the 2001 International Conference on Dependable Systems and Betworks (Formerly: FTCS) (DSN'01). Washington DC, USA: IEEE Computer Society, 2001: 161-172

[28] Hiller M, Jhumkas A, Suri N. Tracking the propagation of data errors in software[M]// Dependable Computing Systems: Paradigms. Performance Issues and Applications, Wiley, 2005: 407-429

[29] Hiller M, Jhumkas A, Suri N. Propane: an environment for examining the propagation of errors in software[J]. Sigsoft Softw. Eng. Notes, 2002, 27(4): 81-85

[30] Li Guo, Gao Jian-min, Gao Zhi-yong, et al. Failure Propagation Model of Complex System Based on Small World Net[J]. Journal of Xi'an Jiaotong University, 2007, 41(3): 334-338(in Chinese)
李果,高建民,高智勇,等. 基于小世界网络的复杂系统故障传播模型[J]. 西安交通大学学报,2007,41(3):334-338

[31] Zhang Fa, Li Lu, Xuan Hui-yu. Survey of Transmission Models of Infectious Diseases[J]. Systems Engineering Theory & Practice, 2011, 31(9): 1736-1744(in Chinese)
张发,李璐,宣慧玉. 传染病传播模型综述[J]. 系统工程理论与实践,2011,31(9):1736-1744

[32] Ni Shun-jiang. Infectious disease dynamics modeling based on complex network theory and research [D]. Beijing: Qinghua University, 2009(in Chinese)
倪顺江. 基于复杂网络理论的传染病动力学建模与研究[D]. 北京:清华大学,2009

[33] Frias-Martinez E, Williamson G, Frias-Martinez V. An Agent-Based Model of Epidemic Spread using Human Mobility and Social Network Information [C]// IEEE Third International Conference on & IEEE Third International Conference on Privacy, Security, Risk & Trust. 2011: 57-64

[34] Ganesh A, Massoulié L, Towsley D. The effect of network topology on the spread of epidemics[C] // Infocom Joint Conference of the IEEE Computer & Communications Societies. IEEE, 2005: 1455-1466

[35] Shafique M, Rehman S, Aceituno P V, et al. Exploiting program-level masking and error propagation for constrained reliability optimization [C]// Proceedings of IEEE/ACM DAC. 2013: 1-9

[36] Zhang J. The calculating formulae, and experimental methods in error propagation analysis [J]. Reliability IEEE Transactions, 2006, 55(2): 169-181

[37] Bhatt D, Schloegel K, Madl G, et al. Quantifying Error Propagation in Data Flow Models[C]// IEEE International Conference & Workshops on Engineering of Computer Based Systems. IEEE Computer Society, 2013: 2-11

[38] Avizienis A, Laprie J, Randell B, et al. Basic concepts and taxono-

my of dependable and secure computing[J]. Publication, 2004, 1(1):11-33

[39] Webster R I, Majnemer A, Platt R W, et al. On failure Propagation in Component-Based Software Systems[J]. Computer & Digital Engineering, 2008, 72(2):492-411

[40] Cortellessa V, Grassi V. Role and impact of error propagation in software architecture reliability[R]. TRCS007/2006. Dipartimento di Informatica, Universita' dell'Aquila, 2006

[41] Cortellessa V, Grassi V. A modeling approach to analyze the impact of error propagation on reliability of component-based systems[M] // Component-Based Software Engineering. 2007: 140-156

[42] Popic P, Desovski D, Abdelmoez W, et al. Error Propagation in the Reliability Analysis of Component Based Systems[C]//Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering. 2005:53-62

[43] Pham T T, Défago X. Reliability prediction for component-based systems: incorporating error propagation analysis and different execution models[C]//Proceedings of 12th International Conference on Quality Software (QSIC12). IEEE Computer Society, Xi'an, China, 2012:106-115

[44] Pham T T, Défago X, Huynh Q T. Reliability prediction for component-based software systems: Dealing with concurrent and propagating errors[J]. Science of Computer Programming, 2015, 97:426-457

[45] Pham T T, Défago X. Reliability prediction for component-based software systems with architectural-level fault tolerance mechanisms[C] // Proceedings of 8th International Conference on Availability, Reliability and Security (ARES13). Regensburg, Germany, 2013:11-20

[46] Fiondella L, Gokhale S S. Architecture-based Software Reliability with Error Propagation and Recovery[C] // International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS). 2013:38-45

[47] Ma Kai. Software fault location method based on the failure propagation research[D]. Dalian: Dalian Maritime University, 2014 (in Chinese)
马凯. 基于故障传播的软件故障定位方法研究[D]. 大连: 大连海事大学, 2014

[48] Wang Yu. Software error locating method based on error propagation context analysis research[D]. Harbin: Harbin University of Industry, 2013(in Chinese)
王煜. 基于错误传播上下文分析的软件错误定位方法研究[D]. 哈尔滨: 哈尔滨工业大学, 2013

[49] Kuck D J, Muraoka Y, Chen S C. On the number of operations simultaneously executable in Fortran-like programs and their resulting speedup[J]. IEEE Transactions on Computers, 1972, 100(12):1293-1310

[50] Ferrante J, Ottenstein K J, Warren J D. The program dependence graph and its use in optimization[J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 1987, 9(3):319-349

[51] Ma P, Wang Y, Su X, et al. A Novel Fault Localization Method with Fault Propagation Context Analysis[C] // International Conference on Instrumentation, Measurement, Computer, Communication & Control. 2013:1194-1199

[52] He Jia-lang, Meng Jin, Zhang Kun, et al. A Fault Propagation-aware Program Fault Location Method[J]. Journal of Electronics & Information Technology, 2011, 33(9):2192-2198(in Chinese)
何加浪, 孟锦, 张琨, 等. 一种故障传播感知的程序故障定位方法[J]. 电子与信息学报, 2011, 33(9):2192-2198

[53] Li Ai-guo, Hong Bing-rong, Wang Si. An approach for Identifying Software Vulnerabilities Based on Error Propagation Analysis[J]. Chinese Journal of Computers, 2007, 30(11):1910-1921(in Chinese)
李爱国, 洪炳镕, 王司. 基于错误传播分析的软件脆弱点识别方法研究[J]. 计算机学报, 2007, 30(11):1910-1921

[54] Voas J. Building Software Recovery Assertions from a Fault Injection-based Propagation Analysis[C]//International Computer Software & Applications Conference. IEEE Computer Society, 1997:505

[55] Suri N, Jhumka A, Hiller M. On the Placement of Software Mechanisms for Detection of Data Errors[C]//Proceedings of the International Conference on Dependable Systems & Networks IEEE Computer Society. 2002:135-144

[56] Dahll G. Error propagation / error containment-a survey[R]. HWR-739 OECD Halden Reactor Project, Halden, 2004

[57] Siewiorek D P, Swarz R S. Reliable Computer Systems: Design and Evaluation[M]. Digital Press, 1992

[58] Treaster M. A Survey of Fault-Tolerance and Fault-Recovery Techniques in Parallel Systems[J]. ACM Research Repository, 2010, 501002:1-11

[59] Schlichting R D, Schneider F B. Fail-Stop Processors: An Approach To Designing Fault-Tolerant Computing Systems[J]. ACM Transactions on Computer System, 1983(3):222-238

[60] Nanya T, Goosen H A. The Byzantine hardware fault model[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1989, 8(11):1226-1231

[61] Bazzi R. Synchronous Byzantine Quorum Systems[J]. Distributed Computing-DC, 2000, 13(1):45-52

[62] Malkhi D, Reiter M. Byzantine Quorum Systems[J]. Distributed Computing, 1998, 11:203-213

[63] Hiller M, Jhumka A, Suri N. Epic: Profiling the propagation and effect of data errors in software[J]. IEEE Transactions on Computers, 2004, 53(5):512-530

[64] Michael C C, Jones R C. On the Uniformity of Error Propagation in Software[C]//Proc. Int'l Conf. Computer Assurance. 1997: 68-76

[65] Johansson A, Suri N. Error propagation profiling of operating systems[C]//Proceedings of the 2005 International Conference on Dependable Systems and Networks. Los Alamitos, CA, USA: IEEE Computer Society, 2005:86-95

[66] Jarboui T, Arlat J, Crouzet Y, et al. Impact of internal and external software faults on the Linux kernel[C]//IEICE Transactions on Information and Systems. 2003:2571-2578

[67] Drebes R J, Nanya T. Analysis of Inter-Module Error Propagation Paths in Monolithic Operating System Kernels[C]//Dependable Computing Conference. European, 2010:175-184

[68] Jia Jia. Study and Implementation of Fault Tolerance for Heterogeneous Parallel Computer[D]. Changsha: National University of Defense Technology, 2011(in Chinese)

(下转第 27 页)

件总数相同的条件下，若新进入队列是微博任务的概率不同，则说明处理微博事件数量占处理总事件数量的比例不同，从而对应最后微博行为的时间间隔分布不同，用个体微博行为的活跃程度机制说明了用户个体行为的时间模式。

由于受到数据采集类型的限制，文中所研究的用户行为模式主要集中在时间特性上。而随着微博平台的不断发展及其功能的不断增加，用户之间的结构性差异也会日趋显现，具有不同年龄背景、身份差异和兴趣爱好的用户会不断增加。在以后的研究中，可以考虑区分个体差异性的影响，查看具有相同身份或兴趣的用户的行为模式是否呈现相同的特点。

参 考 文 献

[1] Barabasi A L. The origin of bursts and heavy tails in human dynamics [J]. Nature, 2005, 435: 207-211

[2] Oliveira J G, Barabasi A L. Darwin and Einstein correspondence patterns [J]. Nature, 2005, 437: 1251-1251

[3] Jiang Z Q, Xie W J, Li M X, et al. Calling patterns in human communication dynamics [J]. Proceedings of the National Academy of Sciences, 2013, 110(5): 1600-1605

[4] Wu Y, Zhou C, Xiao J, et al. Evidence for a bimodal distribution in human communication [J]. Proceedings of the National Academy of Sciences, 2010, 107(44): 18803-18808

[5] Vazquez A, Oliveira J G, Dezso Z, et al. Modeling burst and heavy tails in human dynamics [J]. Physical Review E, 2006, 73(3): 036127

[6] Wu Y, Zhou C, Chen M, et al. Human comment dynamics in online social systems [J]. Physica A: Statistical Mechanics and its Applications, 2010, 389(24): 5832-5837

[7] Wu Y, Ye Q, Li L, et al. Power-law properties of human view and reply behavior in online society [J]. Mathematical Problems in Engineering, 2012(2): 243-253

[8] Kim J, Lee D, Kahng B. Microscopic modelling circadian and bursty pattern of human activities [J]. PloS one, 2013, 8(3): e58292

[9] Song Y, Zhang C, Wu M. The study of human behavior dynamics based on blogosphere [C] // Proceeding of International Conference on Web Information Systems and Mining, 2010. IEEE, 2010: 87-91

[10] Xie J, Zhang C, Wu M. Modeling microblogging communication based on human dynamics [C] // Proceeding of International Conference on Fuzzy Systems and Knowledge Discovery, 2011. IEEE, 2011: 2290-2294

[11] Choi S C, Wette R. Maximum likelihood estimation of the parameters of the gamma distribution and their bias [J]. Technometrics, 1969, 11(4): 683-690

(上接第 9 页)

贾佳. 异构并行计算机容错技术研究[D]. 长沙：国防科学技术大学，2011

[69] Dubrova E. Fault Tolerant Design: An Introduction[M]. Kluwer Academic Publishers, 2006

[70] Somani A K, Trivedi K S. A cache error propagation model [C] // Pacific Rim International Symposium on Fault-tolerant Systems. IEEE, 1997: 15-21

[71] Rashid L, Pattabiraman K, Gopalakrishnan S. Modeling the Propagation of Intermittent Hardware Faults in Programs [C] // Pacific Rim International Symposium on Dependable Computing. IEEE Computer Society, 2010: 19-26

[72] Liu Xin-zhong. Research on Defect Correlation and Its Application[D]. Changchun: Jilin University, 2010(in Chinese)
刘新忠. 关联缺陷及其应用研究[D]. 长春：吉林大学，2010

[73] Jing Tao. An approach for detecting Correlation Software defects[J]. Journal of Software, 2005, 16(1): 17-28(in Chinese)
景涛. 软件关联缺陷的一种检测方法[J]. 软件学报，2005，16(1)：17-28

[74] Bishop P G, Pullen F D. PODS revisited-A study of software failure behavior[C] // Proc. of the IEEE Int'l Symp. on Fault Tolerant Computing. Tokyo, 1988: 1-8

[75] Katerina G P, Trivedi K S. Failure correlation in software reliability models[J]. IEEE Trans. on Reliability, 2000, 49(1): 37-48

[76] Zhang Rong-hui, Jiang Nan, Gou Lang. Software Reliability Growth Model Considering Defect Correlation [J]. Computer Engineering. 2008, 34(8): 44-46(in Chinese)
张荣辉，姜楠，勾朗. 一种考虑缺陷关联的软件可靠性增长模型[J]. 计算机工程，2008，34(8)：44-46

[77] Le W, Soffa M L, Le W. Path-based fault correlations [C] // Proc. of the 18th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. ACM Press, 2010: 307-316

[78] Zhang Da-lin, Jin Da-hai, Gong Yun-zhan, et al. Optimizing Static Analysis Based on Defect Correlations[J]. Journal of Software, 2014, 25(2): 386-399(in Chinese)
张大林，金大海，宫云战，等. 基于缺陷关联的静态分析优化[J]. 软件学报，2014，25(2)：386-399

[79] Sun Jin-shan. Research of Regression Testing Techniques Incorporating Defect Correlation [D]. Changchun: Jilin University, 2010(in Chinese)
孙金珊. 引入关联缺陷的回归测试技术研究[D]. 长春：吉林大学，2010

[80] Wu K, Malaiya Y K. The Effect of Correlated Faults on Software Reliability[C] // International Symposium on Software Reliability Engineering. 1993: 80-89

[81] Dai Y, Xie M, Poh K. Modeling and analysis of correlated software failures of multiple types[J]. IEEE Transactions on Reliability, 2005, 54(1): 100-106

[82] Sue C, Lyu M R, Kuo S, et al. Software Reliability Growth Models Incorporating Fault Dependency with Various Debugging Time Lags[C] // COMPSAC 2004. 2004: 186-191

[83] Zhao Jing, Zhang Ru-bo, Gu Guo-chang. Study on Software Reliability Growth Model Considering Failure Dependency[J]. Chinese Journal of Computers, 2007, 30(10): 1713-1720(in Chinese)
赵靖，张汝波，顾国昌. 考虑故障相关的软件可靠性增长模型研究[J]. 计算机学报，2007，30(10)：1713-1720

[84] Murthy D N P, Nguyen D G. Study of Two-Component System with Failure interaction [J]. Naval Research Logistics Quarterly, 1985, 32(2): 239-247

[85] Li Zhi-rong, Gao Qi, Liu Shen-yang, et al. Reliability Analysis for a Multi-unit System with Failure Rate Interaction[J]. Fire Control & Command Control, 2014, 39(6): 104-107(in Chinese)
栗志荣，高崎，刘慎洋，等. 一种故障相关多单元系统可靠性分析[J]. 火力与指挥控制，2014，39(6)：104-107