基于 BBN 的故障定位技术

柳永坡 金茂忠 贾晓霞 刘雪梅

(北京航空航天大学 计算机学院，北京 100191) (北京城市学院 软件学部，北京 100083)

摘 要：故障定位的目的是帮助程序员寻找引发失效的原因或故障位置，以加快调试过程.故障和失效间的关系往往非常复杂，难以直接描述故障到失效的转化.分析了采用差异分析的方法，提出基于可疑模式，构建故障推理贝叶斯网络，节点由可疑模式及其方法调用者构成；介绍了贝叶斯网络构建算法，各个相关概率的定义及 BBN(Bayesian Belief Network)中各个边的条件概率计算公式.基于推理算法，得到包含故障的模块，并计算得到每个模块包含故障的概率.提出评价方法，并进行了实验验证，取得了平均 0.761 的定准率和 0.737 的定全率，定位结果良好有应用价值.

关 键 词：故障定位；差异；模式发现；概率

中图分类号：TP 311.5

文献标识码：A **文 章 编 号**：1001-5965(2009)10-1201-05

BBN-based fault localization technique

Liu Yongpo Jin Maozhong Jia Xiaoxia

(School of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Liu Xuemei

(College of Software, Beijing City University, Beijing 100083, China)

Abstract: Fault localization techniques help programmers find out the locations and the causes of the faults and accelerate the debugging process. The relation between the fault and the failure is usually complicated, making it hard to deduce how a fault causes the failure. At present, analysis of variance is broadly used in many recent correlative researches. A Bayesian belief network(BBN) for fault reasoning was constructed based on the suspicious pattern, whose nodes consist of the suspicious pattern and the callers of the methods that constitute the suspicious pattern. The constructing algorithm of the BBN, the correlative probabilities, and the formula for the conditional probabilities of each arc of the BBN were defined. A reasoning algorithm based on the BBN was proposed, through which the faulty module can be found and the probability for each module containing the fault can be calculated. An evaluation method was proposed. Experiments were executed to evaluation the fault localization technique. The data demonstrated that 0.761 in accuracy and 0.737 in recall on average were achieved by this technique. It is very effective in fault localization and has high practical value.

Key words: fault location; differentiation; pattern recognition; probability

当软件的实际输出同预期输出不一致时，软件发生失效，而软件失效是由程序中的故障引起的.因此，对故障和失效之间的关系进行分析[1]，是进行故障定位研究的基础.关于故障与失效之间关系的研究，最初可追溯到软件可测试性研究[2].

L. J. Morell[3]于 1984 年 4 月在马里兰大学的一篇技术报告《A Theory of Error-based Tes-

收稿日期：2008-10-21

基金项目：国家自然科学基金资助项目(60603039)

作者简介：柳永坡(1971—)，男，辽宁辽阳人，博士生，liuypo@sei.buaa.edu.cn.

ting》中提出了故障/失效(fault/failure)模型,即故障发生的3个充分必要条件. J. M. Voas[4]提出了PIE(Propagation, Infection and Execution)模型,描述了故障到失效的传播是一个非常复杂的过程. 故障/失效模型和PIE模型都从"故障-失效"的正向角度描述了故障转换为失效的过程. 对于故障定位研究而言,观察到程序失效时,需要从失效出发进行反向分析[5],推断故障的位置或引发故障的原因. 从目前故障定位相关研究的分析可知,故障定位过程中需要使用软件的各种信息,如切片[6]、程序执行轨迹[7]以及基于执行轨迹提取的各种程序行为表示等. 程序执行过程是一个复杂的状态转变过程[8],期间会产生大量的相关信息,使得故障定位成为一个非常复杂的问题.

1 主要故障定位方法比较

表1从确定性/不确定性方法的角度,对已有的故障定位方法进行了对比分析.

表1 采用确定性/不确定性方法的故障定位研究的对比分析

采用方法	技术	定位粒度	优点	缺点
确定性方法	切片/Dice算法调试	变量过程	面向变量询问	规模太大,分析困难 局限性较大
	不变式	不变式	自动分离不变式	无法考虑完整的程序行为
	Tarantula	语句	面向代码	数据量大,且分散
	近邻模型	基本块	分析基本块图谱	数据稀疏
	Delta调试	具体问题	自动运行	运行代价大
不确定性方法	输入级	SIURO	自动运行	只能在输入级
	Dallemeier方法	类	分析方法调用树	只定位在类级别
	Giuseppe判别模式	函数	分析对象行为	没有考虑方法调用的上下文信息

表1中的确定性方法包括切片、Delta调试、利用程序执行轨迹的差异分析技术等[9]. 这些确定性方法在程序本身比较简单的情况下,有很好的效果. 但当程序规模较大,需处理的信息非常多时,方法的效率下降或可用性大大降低. 相比较而言,输入级的故障定位模型[10]、Giuseppe判别模式和Dallemeier方法,这些非确定性方法在程序规模非常大的情况下依然能取得较好的定位结果.

概率推理是指通过一些变量的信息来获取其他概率信息的过程,基于概率推理的贝叶斯网络是为了解决不确定性和不完整性问题而提出的,在多个领域获得了广泛的应用. 文献[11]采用贝叶斯网络对网络中的故障进行诊断;文献[12]利用贝叶斯网络描述复杂设备诊断中存在的不确定性,给出了一种故障诊断方法. 同时,在相关的其他领域,贝叶斯网络也被广泛应用,如基于贝叶斯网络建立软件可靠性早期预测模型[13]等.

本文进行的故障定位研究应用贝叶斯网络进行诊断推理,根据结果推知原因,是自底向上的推理. 包括3个步骤:①确定变量集和变量域;②根据先验知识,确定网络拓扑结构和条件概率表;③运用贝叶斯网络进行故障定位[14].

2 故障推理贝叶斯网络的构建

一般地,构建贝叶斯网络结构的过程包括:利用先验知识构建先验贝叶斯网络;结合已有数据并进行学习,得到后验贝叶斯网络. 本文贝叶斯网络的结构是基于程序失效运行对应的执行轨迹数据建立的,并不需要基于一定的数据进行贝叶斯结构的学习. 下面定义几个相关概念:

定义1 可疑模式集合DPTN(Dubious Pattern):表示待分析程序的所有可疑模式,任一可疑模式 $\mathrm{dptn}=\{m_1\ m_2\ \cdots\ m_i\ \cdots\ m_n \mid m_i \in M\}$,其中 n 为可疑模式dptn的长度,M 为待分析程序中类的所有方法组成的集合.

定义2 调用者集合 C:待分析程序中,所有发出调用请求的方法的集合. 任一调用者 $c \in M$.

定义3 调用边集合 E:调用边为一个有向边,起点为 c,终点为dptn,是指存在方法调用 $c \rightarrow m$,m 为dptn中的任一元素. 这里,dptn与 c 之间是一种推导关系,因此不会出现环状调用情况.

基于上述定义,可以将故障定位的贝叶斯网络定义为三元组 (G,Ω,P). 其中 $G=(V,E)$ 为有向无环图,$V=\mathrm{DPTN}\cup C$;$E=\{(V_i,V_j) \mid V_i \in C, V_j \in \mathrm{DPTN}\}$ 是方法调用边集合,描述变量间的概率依赖关系,Ω 表示域变量的实例化空间,$\Omega=M$. $P=\{\mathrm{CPT}(V_i) \mid V_i \in V\}$ 表示概率分布,即每个域变量的条件概率分布表集合,CPT(Conditional Probability Table)为条件概率表. 图1是用于故障定位的贝叶斯网络的示意图.

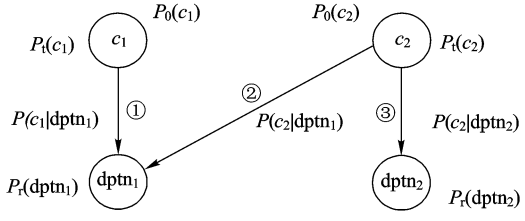


图1 用于故障定位的贝叶斯网络示例图

从图1可以看到,用于故障定位的两层贝叶

斯网络中,底层是可疑模式层,由当前待分析类的所有可疑模式构成,即图 1 中 $dptn_1$ 和 $dptn_2$;而顶层是故障层,为可疑模式的调用者,由构成可疑模式中每个方法的调用者组成,即图 1 中 c_1 和 c_2,且 $dptn_1 \in DPTN$, $dptn_2 \in DPTN$, $c_1 \in C$, $c_2 \in C$,分别标记为①②③的边为调用边,表示了可疑模式层和故障层间的因果关系.

图 1 中,每个节点和边都有相应的概率,本文以节点 c_1, $dptn_1$ 及其间的调用边①为例说明. $P_t(c_1)$表示 c_1 包含故障的概率, $P_r(dptn_1)$表示可疑模式 $dptn_1$ 出现的概率,条件概率 $P(c_1 \mid dptn_1)$表示当可疑模式 $dptn_1$ 出现时,由 c_1 导致其为可疑模式的概率. 为了对条件概率 $P(c_1 \mid dptn_1)$进行计算,定义 $P_0(c_1)$表示 c_1 包含故障的先验概率. 根据贝叶斯公式,其计算过程如下:

$$P(c_1 \mid dptn_1) = \frac{P(dptn_1 \mid c_1) P_0(c_1)}{P_r(dptn_1)} \quad (1)$$

$$P(dptn_1 \mid c_1) = \frac{Occur(dptn_1, FPTN)}{Occur(ptns, c_1)} \quad (2)$$

式(1)中, $P_r(dptn_1)$为一个联合概率,为 $dptn_1$ 出现在模式集中的概率同其出现在可疑模式集中的概率的乘积,即 $\frac{Occur(dptn_1, PTN)}{|PTN|}$ 和 $\frac{Occur(dptn_1, FPTN)}{Occur(dptn_1, PTN)}$.

式(2)中, $Occur(dptn_1, FPTN) = Support[dptn_1 \text{ in } FPTN]$,是指 $dptn_1$ 出现在失效运行对应的类的被调用方法序列集的次数; $Occur(ptns, c_1) = |m_i \text{ in } ptn_k|$,是指当 c_1 作为方法的调用者时,分析程序运行轨迹获取 c_1 调用过的所有方法,并累积这些方法出现在程序全部运行的模式集中和 $dptn_1$ 长度相等的模式中的次数. 而 $ptn_k \in \{ptn_i \mid ptn_i \in SPTN \cup FPTN\}$, $m_i \in ptns$, m_i 被 c_1 调用且 $len(ptn_i) = len(dptn_1)$, FPTN(Failure Pattern)表示失效模式集, SPTN(Suspicious Pattern)表示成功模式集.

计算得到的结果就是图 1 中边 $(c_1, dptn_1)$的概率意义,表示当 $dptn_1$ 为一个反常行为时, c_1 导致其反常的概率. 类似地,可以计算得到边 $(c_2, dptn_1)$的概率意义 $P(c_2 \mid dptn_1)$.

对待分析程序的每一个类构造一个贝叶斯网络,在同一个贝叶斯网络的节点中,任意 $c_i = c_j$, $dptn_i \neq dptn_j$,也就是说,同一节点只允许出现一次. 构建类的贝叶斯网络从分析类的可疑模式开始,下面是构造 BBN(Bayesian Belief Network)结构算法(时间复杂度为 $O(n^2)$):

buildBBN(DPTN)

```
Input: DPTN, Output: a two-layer BBN
{
  For each pattern dptn in DPTN{
    dptn_node = buildptnNode(dptn);
    For each method m in dptn{
      m_caller = m.caller;
      check = checkExist(m_caller);
      if(check == TRUE){
        ptn_node = findNode(m_caller);
        buildConn(m_caller, dptn_node);
    }else{
        m_node = buildMethodNode(m_caller);
        buildConn(m_node, dptn_node);
    }}}}}
```

该算法分析类的可疑模式集,每个可疑模式构成贝叶斯网络的低层节点. 依次对每个可疑模式进行分析,对构成的每个方法,依据程序执行轨迹获取调用者,调用者节点为网络中的高层节点,在调用者和包含被调用者的可疑模式间增加一条调用边,建立形如图 1 中的贝叶斯网络. 这里,类的所有可疑模式及构成每个可疑模式的方法的调用者,构成贝叶斯网络的节点,可疑模式与其调用者间的调用边构成贝叶斯网络中的有向边,贝叶斯网络中的节点和边被创建.

3　基于 BBN 推理网络的求解过程

对故障定位问题而言,从正向角度来看,由于可疑模式中方法的调用者 c 可能包含故障,从而导致可疑模式的出现, c 包含故障是原因, dptn 的出现是结果;从逆向角度,也就是从故障定位的角度来看,可疑模式的出现表示程序行为出现了反常,发现可能包含故障的模块(也就是调用者)时, dptn 是"起因", c 是结果. 因此,可以基于已经建立的 BBN 对故障进行推理. BBN 包含方法层和可疑模式层. 对方法层中的每个节点 c 而言,发出的方法调用导致(或部分导致)了可疑模式的出现. 也就是说, c 是导致可疑模式出现的原因. 如果程序员把程序失效归结为可疑模式的出现,可疑模式就可以把程序中的失效归结为(调用自己的)方法中包含的故障. 第 1 个归结是通过检测可疑模式进行的,而第 2 个归结便通过 BBN 来完成.

对贝叶斯网络中方法层的每个节点 c,定义与其相关的可疑模式集合为 c_DPTN,则 c 包含

故障的概率 $P_t(c)$可以通过下式计算：

$$P_t(c)=\sum_{\text{dptn in } c_DPTN} P(c \mid \text{dptn}) \times P_r(\text{dtpn}) \quad (3)$$

计算得到一个类对应的贝叶斯网路中的每个调用者节点 c，可能包含故障的概率 $P_t(c)$. 依据 $P_t(c)$对节点进行排序，便是初始故障定位结果，而初始定位结果中的每个模块并不一定都包含故障. 依照前文分析，并不真正包含故障的模块，其故障概率相对较小. 因此，可以依据 $P_t(c)$对得到的故障模块进行过滤，将概率值相对较小的模块舍弃，并对过滤后的故障模块集中每个模块的故障概率进行规一化(normalization).

设 CF 表示某个类经贝叶斯分析后得到的故障模块集合，对于其中的任意故障模块 cf_i，$P_t(cf_i)$表示 cf_i的故障概率，$\min_P_t$为 CF 中所有模块故障概率的最小值，$\min_P_r=\min\{P_t(cf_i)\}$ $(1\leqslant i\leqslant |CF|)$，定义 cf_i的保留度为

$$rd(cf_i)=\frac{P_t(cf_i)}{\min_P_t} \quad (4)$$

定义 β 为保留度阈值，由人工设置. 如果 $rd(cf_i)>\beta$，则认为 cf_i包含故障，予以保留；否则从 CF 中去除. 这样，经过过滤后得到的故障模块集称为 RCF. 对 RCF 中的每个 cf 进行规一化：

$$P_t(cf_i)=\frac{P_t(cf_i)}{\sum P_t(cf_i)} \quad (5)$$

依据 $P_t(cf_i)$对 RCF 中的模块进行排序，得到最终的故障定位结果.

4 结果对比分析

将本文阐述的故障定位方法与 Dallmeier[15-16] 的故障定位方法进行了对比分析，如表 2.

表 2 本文方法和 Dallmeier 方法的对比分析

分析因素	本文方法	Dallmeier 的方法
模式分析	不连续模式分析，序列模式挖掘	连续模式分析，滑动窗口的算法获得序列集
图谱类型	count 图谱，分析模式出现的次数	hit 图谱，分析序列是否出现
故障推理	贝叶斯网络	定义类包含故障的权值
定位级别	模块	类
评价方法	定准率和定全率	搜索长度

表 2 从 5 个不同的方面进行了对比分析，2 种方法都基于对象方法调用轨迹表示对象行为，采用差异分析方法进行故障定位. Dallmeier 分析对象方法的被调用序列及调用序列，认为调用序列更易于揭示类中可能包含的故障，即只观察失效运行和成功运行对应的序列集间的差异，没有进行进一步的推理. 本文方法分析了对象的被调用方法序列并发现可疑模式，然后分析程序执行轨迹进行反向推理，从观察到的现象找到产生原因.

在故障推理方面，Dallmeier 通过分析序列是否出现在失效运行和成功运行对应的序列集中，计算类包含故障的权值. 本文方法基于可疑模式，采用贝叶斯网络进行推理，得到了可能包含故障的模快. 在定位级别上，Dallmeier 给出了可能包含故障的类的排列，本文方法给出了可能包含故障的模块(类的方法)序列，定位粒度更小. 此外，本文方法采用了定准率和定全率对计算结果进行了评价，而 Dallmeier 则依据搜索长度——结果排列中故障类之前的类的个数，分析得到真正包含故障的类.

因此，本文基于对象行为信息进行了更为深入的分析，并对程序行为间异常进行了反向推理，故障定位粒度也更为精细.

5 实验验证

本文选取了 7 个真实软件并进行 2 组实验. 第 1 组包括 DataTest，ThreadTest，FileChooserDemo，Maze，DecimalToBinary 以及 AutoMachine. 第 2 组包括 WSDL. 对第 1 组软件进行受控实验，对第 2 组软件进行非受控实验. 分别运行并收集了 3 次正确运行和 3 次失效运行的程序执行轨迹，基于这些数据计算得到故障定位结果，获得了平均 0.761 的定准率和平均 0.737 的定全率. 下面对定位的实验结果进行分析，表 3 列出了本文的故障定位方法在 2 组共 11 个实验基础上的性能及其他相关的指标数据.

表 3 程序定位性能表

Software	Precision	Recall
DataTest	0.83	1
ThreadTest	0.83	1
DecimalToBinary2	0.78	0.88
DecimalToBinary1	0.71	0.71
Maze	0.83	0.71
FileChooserDemo	0.75	0.67
AutoMachine1	0.75	0.64
AutoMachine2	0.72	0.62
AutoMachine3	0.72	0.58
AutoMachine4	0.7	0.55
WSDL	0.75	0.75
平均值	0.761	0.737

表 3 列出了待分析程序的平均定位性能，可以看到，针对 Precision 指标而言，除了 AutoMachine3 和 AutoMachine4，其余的都有不错的效果. 对 Recall 指标而言，AutoMachine3 和 Auto-

Machine4 的效果也相对低一些.但对定位结果进一步分析(略)可知,方法可能包含故障的概率值的相对大小,可以帮助调试者更有效地进行下一步分析.

表 4 中给出了几种典型的程序复杂性度量值,是采用经典的度量方法对实验中程序和类的复杂性进行度量得到的.这些数据支持了相关领域的 3 个观点:程序越复杂,就越难找到其中隐藏的故障;复杂度高的类所隐藏的故障,比复杂度低的类中所隐藏的故障更难发现;程序故障密度越大,发现故障的难度就越小.

表 4　程序复杂度指标表

Software	ABD	AC	WM	EC	Difficulty	FD
DataTest	1.25	1.25	15	1	11.55	70.42
ThreadTest	1.06	1.66	25	4	21.58	57.47
DecimalToB2	0.82	1.03	29	5	10.41	66.67
DecimalToB1	0.82	1.03	29	5	10.46	58.33
Maze	0.75	1.36	79	9	51.46	25.55
FileChooserD	0.91	1.98	147	6	59.64	14.85
AutoMach1	1.4	2.63	512	17	196.42	5.78
AutoMach2	1.4	2.63	512	17	196.31	5.37
AutoMach3	1.4	2.63	512	17	196.31	4.95
AutoMach4	1.4	2.63	512	17	196.39	4.54
WSDL	1.02	1.52	201	6	41.26	5.87

注:ABD—Average Block Depth;AC—Average Cyclomatic;WM—Weighted Method;EC—Efferent Coupling;FD—Fault Density.

6　结 束 语

本文基于可疑模式,建立用于逆向推理的贝叶斯网络,构建了故障推理模型.包括贝叶斯网络结构的建立、概率计算以及如何基于贝叶斯网络推理,获得可能包含故障的方法及其概率.最后通过实验,获得了平均 0.761 的定准率和平均0.737 的定全率.

参考文献(References)

[1] Andreas Z, Ralf H. Simplifying and isolating failure-inducing input[J]. IEEE Transactions on Software Engineering, 2002, 28(2):183－200

[2] Mary J H, Gregg R, Kent S, et al. An empirical investigation of the relationship between spectra differences and regression faults [J]. Journal of Software Testing, Verification, and Reliability, 2000, 10(3):171－194

[3] Morell L J. A theory of fault-based testing[J]. IEEE Transactions on Software Engineering, 1990,16(8):844－857

[4] Voas J M. PIE: a dynamic failure-based technique[J]. IEEE Trans Software Eng, 1992, 18(8): 717－727

[5] James A J, Mary J H, John S. Visualization of test information to assist fault localization[C] // 24th International Conference on Software Engineering (ICSE 2002). Florida: ACM, 2002:467－477

[6] Shinji K, Akira N, Keisuke N, et al. Experimental evaluation of program slicing for fault localization[J]. Empirical Software Engineering, 2002 ,7(1): 49－76

[7] Margaret A F. Fault localization through execution traces [D]. USA: Georgia Institute of Technology, 2002

[8] Manos R, Steven P R. Fault localization with nearest neighbor queries[C] //18th IEEE International Conference on Automated Software Engineering (ASE 2003). Montreal: IEEE Computer Society, 2003:30－39

[9] James A J, Mary J H. Empirical evaluation of the tarantula automatic fault-localization technique[C] // 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005). California: ACM, 2005:273－282

[10] Wu Ji, Jia Xiaoxia, Liu Chang, et al. A statistical model to locate faults at input level[C] //IEEE 19th International Conference on Automated Software Engineering (ASE 2004). Linz: IEEE Computer Society, 2004:274－277

[11] 汪成亮,沈文武,张勤,等.以因果网为理论核心的网络故障诊断方法[J].计算机工程与应用,2004,40(4):151－153
Wang Chengliang, Shen Wenwu, Zhang Qin, et al. Computer network fault diagnosis based on causality diagram theory[J]. Computer Engineering and Applications, 2004, 40(4):151－153(in Chinese)

[12] 李俭川.基于贝叶斯网络的智能故障诊断方法[J].中国惯性技术学报,2002,10(4):24－28
Li Jianchuan. Intelligent fault diagnosis technique based on bayesian networks [J]. Journal of Chinese Inertial Technology, 2002,10(4):24－28(in Chinese)

[13] 汪浩.基于数据挖掘技术的软件可靠性模型研究[D].北京:北京航空航天大学计算机学院,2002
Wang Hao. Research of software reliability model based on data mining[D]. Beijing: School of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, 2002(in Chinese)

[14] Murphy K P. A brief introduction to graphical models and bayesian Networks [EB/OL]. http://www.ai.mit.edu/~murphyk/Bayes/bayes.html

[15] Valentin D, Christian L, Andreas Z. Lightweight defect localization for java[C] //The 19th European Conference on Object-Oriented Programming (ECOOP 2005). Glasgow, Scotland: Springer, 2005:528－550

[16] Valentin D, Christian L, Andrzej W, et al. Mining object behavior with ADABU[C] //The IEEE Fourth International Workshop on Dynamic Analysis (WODA 2006). Shanghai: ACM, 2006:17－24

(编　辑:赵海容)