一种基于信息熵的多维流数据噪声检测算法

李文忠[1] 左万利[2] 赫枫龄[2]
(吉林大学计算机科学与技术学院 长春 130012)[1]
(吉林大学符号计算与知识工程教育部重点实验室 长春 130012)[2]

摘 要 流数据噪声检测是一个比较困难的领域。由于流数据的一些特殊性,使得以前的一些处理静态数据的算法对于流数据的处理而言都不理想。在局部离群点(LOF)思想的基础上,引入信息熵来计算数据各维属性的信息增益,并根据各维属性的增益来计算数据的局部离群度,提出一种多维流数据数据噪声点检测算法EDLOF。实验结果表明,该算法对于多维流数据有较好的适应性,而且具有一定的泛化能力。

关键词 噪声检测,流数据,数据挖掘,信息熵
中图法分类号 TP311 **文献标识码** A

Entropy-based Algorithmin for Noise Detection in Multi-dimensional Stream Data

LI Wen-zhong[1] ZUO Wan-li[2] HE Feng-ling[2]
(College of Computer Science and Technology, Jilin University, Changchun 130012, China)[1]
(Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China)[2]

Abstract Noise detection in stream data is a little more difficult area. Most of the algorithms used to deal with the static data are not helpful to process the stream data any more. Considering the idea of local outlier, we used entropy to measure the outliers and then the algorithm EDLOF was proposed to deal with the multi-dimensional stream data. And the results of experiments show that it is useful to process multi-dimensional of stream data, and it gets generalization ability at the same time.

Keywords Noise detection, Stream data, Data mining, Information entropy

1 研究背景

噪声数据检测是数据挖掘的一个重要方面,对于数据处理结果的准确性和精度有着重要的影响。

对于给定的一个数据集合,其中的某些对象在属性值上可能与数据集整体的分布行为不一致,这些少量的区别于其它大多数的对象的对象,一般就称为噪声点。噪声点,又被称为异常点,离群点或孤立点。Hawkinsl 最早给出了噪声点的本质性定义:噪声点如此不同于数据集中的其它数据,以至于使人怀疑这些数据并非随机偏差,而是产生于完全不同的机制[1]。

传统的数据,一般存储在关系数据库中,是一种静态的数据。流数据不同于传统的数据,而是一种实时的、动态的数据,即随着时间的变化,流数据会随着增长。对比传统的静态数据,流数据具有以下几个特点[2]:1)数据实时到达;2)数据到达次序独立,不受应用系统所控制;3)数据规模宏大且不能预知其最大值;4)数据一经处理,除非特意保存,否则不能被再次取出处理,或再次提取数据代价昂贵。

流数据的这些特点决定了流数据挖掘算法必须具有如下特点:(1)时间复杂度低。流数据实时的变化,要求算法的处理速度不能低于流数据到达的速度,否则会阻塞流数据。(2)空间复杂度低。由于内存资源的限制,不可能一次装入全部的数据到内存中。另外由于流数据挖掘的高效率要求,也决定了一般不可能对流数据进行转存储处理。(3)适应性。流数据是一种动态的数据,数据集一般不能同时到达,而且到达的次序也不确定,这就要求算法具有一定的适应性,不能受数据集规模和输入次序的影响。

静态数据噪声挖掘算法大致可以分为以下几种类型[6]:a)基于统计学的方法,包括基于分布的方法、基于深度的方法。b)基于距离的方法,包括基于索引的算法、嵌套-循环算法、基于单元的算法。c)基于密度的方法。d)基于聚类的方法。e)基于偏离的方法,包括序列异常技术、OLAP数据立方体技术。

流数据中的噪声点不同于传统静态数据中的噪声点,它

到稿日期:2011-03-28 返修日期:2011-07-17 本文受国家自然科学基金(60973040),国家自然科学青年基金(60903098),吉林省科技发展计划项目(20070533),教育部高等学校博士学科点专项科研基金(200801830021)和吉林大学基本科研业务费交叉学科与创新项目(200810025)资助。

李文忠(1986-),男,硕士生,主要研究方向为数据挖掘和Web搜索引擎,E-mail:liwenzhong1016@126.com;左万利(1957-),男,教授,博士生导师,主要研究方向为数据库理论、机器学习、数据挖掘、Web挖掘以及搜索引擎;赫枫龄(1962-),男,硕士,教授,主要研究方向为Web挖掘和搜索引擎。

更多地是从局部来定义。Markus M. Breunig 在文献[3]根据局部离群点思想提出了一种基于局部离群度的检测算法 LOF；倪巍伟、陈耿等人在文献[4]中，通过分析局部信息熵和子空间，提出了检测算法(SPOD)；于绍越、商琳在文献[5]中提出了去一划分信息增量的概念，并基于此提出了 ENBROD 算法。薛安荣、鞠时光等，在文献[7]中提出了空间局部离群系数 SLOF，及基于 SLOF 的空间离群点挖掘算法。

目前流数据挖掘中，一般都是应用滑动窗口来划分流数据，然后对这些窗口的数据进行挖掘，例如张月琴在文献[9]中采用频繁项集的方法。曾颖等在文献[10]中，通过对流数据进行划分先进行 K-Means 聚类，然后进行聚合聚类的流数据噪声检测算法。毛国君等在文献[11]中根据数据的多维频度特点，设计出 MaxFP-Tree 数据结构，提出算法 MaxFPinNDS。倪巍伟等人，在文献[12]中提出了 DSOKP 算法，该算法通过对流数据进行分区，在每个分区内基于 K-Means 聚类获得均值点集，然后识别出噪声数据。

2 信息熵以及相关定义

信息熵最早(1908)由 Claude E. Shannon 提出，用来度量系统的不确定性程度。当熵值较小时，表明该系统越稳定、纯净，反之表明系统越不稳定、越杂乱[8]。参照 Hawkinsl 给出的噪声数据定义，噪声数据的存在会使得数据集不纯净。可以利用信息熵来进行数据去噪。

设有随机变量 $X=\{x_i\}$，$i=1,2,\cdots,n$，且其概率密度函数为 $p(x)$，则其熵定义为 $Entropy(X)=\sum_{i=1}^{n}-P(x_i)\log_a 1/p(x_i)$，其中 a 取自然数，一般常取 $a\in\{2,e,10\}$。取 $a=2$ 时，熵的单位为 bit；取 $a=e$ 时，熵的单位为 nat；而当 $a=10$ 时，熵的单位为 dit。

特别地，对于随即变量 $X=(x_1,x_2,\cdots,x_n)$，其中 $x_i(i=1,2,\cdots n)$ 也为随即变量，那么有 $Entropy(X)=\sum -p(x_1,x_2,\cdots,x_n)\log_a(1/p(x_1,x_2,\cdots,x_n))$，当 $x_1,x_2,\cdots,x_n$ 相互独立时，有 $Entropy(X)=\sum_{i=1}^{n}Entropy(x_i)$。

设有 n 维数据集 $D=\{d|d\in(X_1,\cdots,X_n,Label)\}$，其中 $X_1,\cdots,X_n$ 为常规属性，属性 $Label$ 为决策属性，标示数据的分类，且设 $Label=\{l_1,l_2,\cdots,l_c\}$。

定义 1(集合 D 的熵 $Entropy(D)$)

$$Entropy(D)=\sum_{i=1}^{c}-p_i\log_a p_i \tag{1}$$

式中，p_i 表示集合 D 中属于 c_i 类的比例，并且此处取 $a=2$。

定义 2(属性 X_n 关于集合 D 的信息增益熵 $Gain(D,X_n)$)

设集合 D 中数据在属性 X_n 上的取值情况为 $\{(S_v,v)\}$，其中 $S_v=\{s\in S|X_n(s)=v\}$，即 S_v 表示在 X_n 维上取值为 v 的数据集合。那么

$$Gain(D,X_n)=Entropy(D)-\sum_{v}\frac{|S_v|}{|D|}Entropy(S_v) \tag{2}$$

在不引起歧义的情况下，一般简记 $Gain(D,X_n)$ 为 $Gain(X_n)$。

定义 3(向量 x,y 的差熵 $\mathrm{Sim}(x,y)$)

设有向量 $x,y\in\{d|d=(x_1,\cdots,x_n)\}$，那么

$$\mathrm{Sim}(x,y)=\sum_{i=1}^{n}Gain(X_i)\,|x_i-y_i| \tag{3}$$

定义 4(向量 x 关于集合 D 的离群熵 $Dis(D,x)$)

设 $x\in D$，记集合 $D^*=D-\{x\}$，那么

$$Dis(D,x)=\sum_{\forall y\in D^*}\mathrm{Sim}(x,y) \tag{4}$$

在不引起歧义的情况下，一般简记 $Dis(D,x)$ 为 $Dis(x)$。

3 噪声检测算法

离群熵是基于这样一个事实，在给定的数据集合 S 中，设噪声数据集为 O，则正常数据集为 $S^*=S-O$。分别取正常数据集 S^* 和噪声数据集 O 中的数据来计算离群熵。由于 D^* 中的正常数据之间更为相似，而 O 中的数据则偏离正常数据集 D^*，并且正常数据集一般都远远大于离群数据集，即 $|D^*|>>|O|$，因此计算得到的离群熵会有很大不同，这些有着明显不同于绝大多数的数据就可以被认为是噪声数据。计算各个数据相对于数据集合的离群熵首先需要获得数据每一维属性的信息增益，而通过对样本进行训练可以准确地获取各个属性的增益。即在进行噪声检测之前，先对一个包含正常数据和噪声数据的训练集进行分析，从而获得各个属性的信息增益。

本文设计的流数据噪声清除过程如图 1 所示。

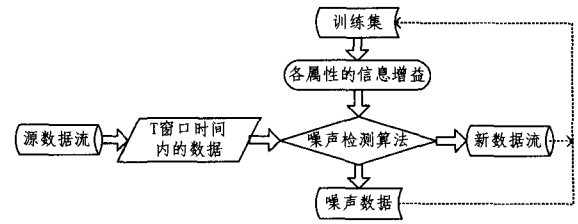

图 1 噪声监测框图

3.1 算法思想

首先根据训练数据集来获得各维属性的信息增益，然后依次读取源数据流 T 时间内的数据，记为集合 D，且 $|D|=N$，即相当于窗口大小为 N；依据本文给出的噪声点检测算法，进行噪声点检测，从源数据中剔除噪声点数据，输出剔除噪声点后的数据流。同时由反馈的思想，可以根据检测出的噪声数据和新数据流中的正常数据来重新构造训练集，以获得更准确的属性增益值。

3.2 EDLOF 算法描述

输入：源数据流 S，初始训练集 L_0

输出：噪声数据

Step1 依据式(1)，式(2)计算获得 L_0 中各维属性的增益。

Step2 从数据流上读取时间窗口为 T 内的数据，不妨设为 D。

Step3 $\forall x\in D$，根据式(3)，式(4)计算 $Dis(D,x)$；

Step4 对计算出的 $Dis(D,x)$ 进行降序排序，得到序列 $\{h_1,h_2,\cdots,h_n\}$，分离出噪声数 D^*；

Step5 如果数据流 S 未处理完，则分别随机取 D^* 和 $D-D^*$ 中的部分数据加入到 L_0，即 $L_0=L_0\cup\alpha D^*\cup\beta(D-D^*)$，然后转 Step1，直到数据流 S 处理完。

在 Step4 中分离出离群点，既可以简单地取序列中具有最大离群熵的前 k 个为噪声数据，还可以计算出该序列的 Δ 差，即 $\Delta_i=h_{i+1}-h_i$，$1\leqslant i<n$。噪声数据 Δ_i 的变化大于正常数据的，所以可以找到相邻 2 个 Δ_i 变化较大的，设为 Δ_k，Δ_{k+1}，那么就可以认为前 k 个，即 $h_1,h_2,\cdots,h_k$ 对应的数据为噪声数据。

Step5 中，当检测出的噪声数据集 D^* 较小时，可以考虑全部加入训练集 L_0，即 $\alpha=1$。一般情况下可取 $\alpha\in[0.3,1]$，$\beta\in(0,0.1)$。

3.3 算法的时空复杂度分析

3.3.1 时间复杂度

EDLOF 算法需要扫描一遍数据，对于每一个向量 x，需要计算它与其他所有向量 $y\in D-\{x\}$ 的 $\mathrm{Sim}(x,y)$。考虑到 $\mathrm{Sim}(x,y)=\mathrm{Sim}(y,x)$，设有 $|D|=n$，则需要计算 $n*(n-1)/2$ 次，故时间复杂度为 $O(n^2)$。

3.3.2 空间复杂度

算法执行过程中只需要储存每次所处理的数据，并额外保存每一维属性的增益，以及每个向量的 $Dis(x,D)$，即空间复杂度为 $O(n)$。

4 试验以及分析

测试数据集来自 UCI 的 Zoo 数据集[13]和 UCI 的 CUP99 数据集[14]。实验环境：Intel Core2 Dual 2.6G，2G RAM，Windows7 Professional，在 Visual Studio 2010 上用 C/C++语言实现。

4.1 实验 1

采用 Zoo 数据集。Zoo 数据集中包含了 101 种动物的统计数据记录，并且把这 101 个动物分成了 7 大类。每个记录包含了 18 个属性，其中 17 个常规属性，1 个决策属性。17 个常规属性包括：1 个动物名称，15 个布尔属性，1 个离散数值属性(腿的个数)。1 个决策属性表明所属的种类。分类数据如表 1 所列。

表 1 Zoo 数据集

种类	总数	动物名
S1	41	aardvark, antelope, bear, boar, buffalo, calf…
S2	20	chicken, crow, dove, duck, flamingo, gull, hawk…
S3	5	pitviper, seasnake, slowworm, tortoise, tuatara
S4	13	bass, carp, catfish, chub, dogfish, haddock…
S5	4	frog, frog, newt, toad
S6	8	flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp
S7	10	clam, crab, crayfish, lobster, octopus, …

试验中选取 S_1 类中的前 30 个数据，记为集合 l_1，取 S_3 类中的前 3 个数据记为 l_3。

训练数据集合为：$l=l_1\cup l_3$，训练时，以 l_3 为噪声数据进行训练，l_1 为正常数据。

(1)测试数据集 $T_1=S_1\cup S_3$，去除其中的类别属性，由于数据规模较小，因此在一次窗口时间内读入全部数据，且反馈系数 $\alpha=\beta=0$。与 ENBROD[5]进行对比，其中 MinPts=30，MitPts=46，实验结果如表 2 所列。

表 2 T_1 噪声检测结果

	噪声个数	噪声数据
ENBROD, MinPts=30	5	seasnake, tortoise, pitviper, seal, fruitbat
ENBROD, MinPts=46	5	seasnake, tortoise, pitviper, slowworm, seal
EDLOF	8	seasnake, tortoise, pitviper, slowworm, tuatara, seal, porpoise, dolphin

(2)测试数据集 $T_2=S_1\cup S_3\cup S_5$，去除其中的类别属性，在一次窗口时间内读入全部数据，且反馈系数 $\alpha=\beta=0$。与 ENBROD[5]进行对比，其中 MitPts=50。实验结果如表 3 所列。

表 3 T_2 噪声检测结果

	噪声个数	噪声数据
ENBROD, MinPts=50	6	seasnake, tortoise, pitviper, slowworm, seal, frog
EDLOF	12	pitviper, seasnake, slowworm, tortoise, tuatara, seal, porpoise, dolphin, frog, frog, newt, toad

由表 2 可以看出，EDLOF 算法能有效地检测出 S_3 中的所有目标噪声数据，而 ENBROD 算法在 MitPts=46 时只能检测出 S_3 中的 4 个，该算法没有能够识别出 tuatara。另外，EDLOF 算法还检测出 S_1 类中的 seal，porpoise，dolphin，认为它们虽然属于 S_1 类，但是还是偏离 S_1 中的其它动物。同样地，ENBROD 算法检测出了 S_1 中的 seal。

由表 3 看出，EDLOF 算法具备较强的泛化能力，虽然训练数据集 $l=l_1\cup l_3$ 中并不包含 S_5 类中的数据，但是除了检测出 S_3 类中的数据外，还能完整地检测出测试集 T_2 中的 S_5 类数据。ENBROD 算法识别出了 S_5 类中的一个数据 frog。需要说明的是，S_5 类中的数据包含两条 frog 的记录，它们属性值不一样，不是重复的记录。

上面试验中测试数据集规模小，说明 EDLOF 算法对于数据的规模没有要求，对小规模的数据也有很好地适应能力。

4.2 实验 2

采用 CUP99 数据集。CUP99 数据集中每个记录包含 41 个常规属性和 1 个决策属性。这个决策属性表明了攻击的种类。

试验中取训练集中的 800 条数据来进行训练，其中包括 750 条正常记录(决策属性值为 normal)，40 条 R2L 类下的 snmpguess 类攻击记录，10 条 R2L 类下的其它类的攻击记录。对于记录中的离散型非数值变量，比喻协议类型，服务类型等，从 1 开始进行编号。对于数值类型的连续变量，比如创建文件次数、访问 root 次数等，先划分区间，然后离散化。

测试数据集取自 Correct 中的 5000 条记录，其中包含有 200 条 snmpguess 类攻击记录(4%)，20 条 R2L 类下的其它类型的攻击记录(0.4%)，即噪声数据为 220，占总数据的 4.4%。记源数据流中的噪声数据为集合 S，则 $|S|=220$。

EDLOF 算法中，窗口规模为 500 个记录。DSOKP 算法中，$k=6$，分块规模为 500。算法对比主要分为两个方面：一方面是检测出的准确率 p 和错误率 q；另一方面是算法的执行时间。

设检测出的噪声数据集为 D，那么，检测正确率 $p=\frac{|S\cup D|}{|S|}$，检测错误率为 $q=\frac{|D-S|}{N}$，其中，N 为源数据流中正常数据的数量。由于 $|N|=4780$，导致 q 将会是一个很小的值，因此统计时用错误个数代替，即用 $|D-S|$ 来代替。实验结果如表 4 所列。

表 4 检测结果

	噪声个数	正确个数	正确率	错误个数	执行时间
DSOKP	197	193	87.7%	4	1343s
EDLOF	223	187	85%	36	419s

由以上实验可以看出，EDLOF 算法检测出的个数多于 DSOKP 算法，但其正确率比 DSOKP 算法略差，而且检测结果中出现了较多的属于正常的数据。这有可能是由于在训练时出现了数据的过度拟合。在执行效率方面，DSOKP 算法中对于每个分区的数据采取的是 K-Means 算法，执行时间较

长。

结束语 本文针对多维流数据，提出了一种基于信息熵的噪声检测算法 EDLOF。依据局部噪声点的思想，在数据属性的信息增益的基础上，给出了数据点的局部离群熵的定义，并根据数据点的局部离群熵寻找到流数据中的噪声数据。实验 1 在 UCI Zoo 数据集上将 EDLOF 算法与 ENBROD 算法进行对比，实验表明 EDLOF 算法对噪声数据的检测有较高的准确性，而且对数据的规模没有要求，并且其对于训练集中未出现的未知噪声数据类型有着较好的泛化能力。实验 2 在 UCI CUP99 数据集上将 EDLOF 算法与 DSOKP 算法进行对比，实验表明在正确率相差不大的情况下，由于数据的过度拟合等原因，EDLOF 算法检测出的正常数据比 DSOKP 算法检测出的多，但是 EDLOF 算法有着更高的执行效率。

参 考 文 献

[1] Hawkins D. Identification of Outliers[M]. Chapman and Hall, London: Chapman and Hall, 1980

[2] 金澈清，钱卫宁，周傲英. 流数据分析与管理综述[J]. 软件学报，2004，15(8)：1172-1181

[3] Breunig M M, Kriegel H-P, Ng R T, et al, LOF: Identifying Density-Based Local Outliers[A]//Proc. ACM SIGMOD 2000, Dallas[C]. ACM Press. New York: ACM, 2000: 93-104

[4] 倪巍伟，陈耿，陆介平，等. 基于局部信息熵的加权子空间离群点检测算法[J]. 计算机研究与发展，2008，45(7)：1189-1194

[5] 于绍越，商琳. 基于信息熵的相对离群点的检测方法：ENBROD[J]. 南京大学学报：自然科学版，2008，44(2)：212-218

[6] 徐翔，刘建伟，罗雄麟. 离群点挖掘研究[J]. 计算机应用研究，2009，26(1)：34-40

[7] 薛安荣，鞠时光. 局部离群点挖掘算法研究[J]. 计算机学报，2007，30(8)：1455-1463

[8] 熊家军，李庆华. 信息熵理论与入侵检测聚类问题研究[J]. 小型微型计算机系统，2005，26(7)：1163-1166

[9] 张月琴. 滑动窗口中数据流频繁项集挖掘方法[J]. 计算机工程与应用，2010，46(16)：132-134

[10] 曾颖，罗可，邹瑞芝. 基于 K-均值聚类和凝聚聚类的离群点查找方法[J]. 计算机工程与应用，2009，45(26)：131-133

[11] 毛国君，宗东军. 基于多维数据流挖掘技术的入侵检测模型与算法[J]. 计算机研究与发展，2009，46(4)：602-609

[12] 倪巍伟，陆介平，陈耿，等. 基于 k 均值分区的数据流离群点检测算法[J]. 计算机研究与发展，2006，43(9)：1639-1643

[13] http://archive.ics.uci.edu/ml/datasets/Zoo, http://archive.ics.uci.edu/ml/machine-learning-databases/zoo/

[14] http://kdd.ics.uci.edu/databases/kddcup99/

(上接第 178 页)

依据以上的系统组成表和重构信息表，可以分别作出系统模态层、使命模态层以及功能模态层的重构蓝图规划。以表 1 中的系统组成为例，得到的系统模态规划蓝图如图 8 所示。

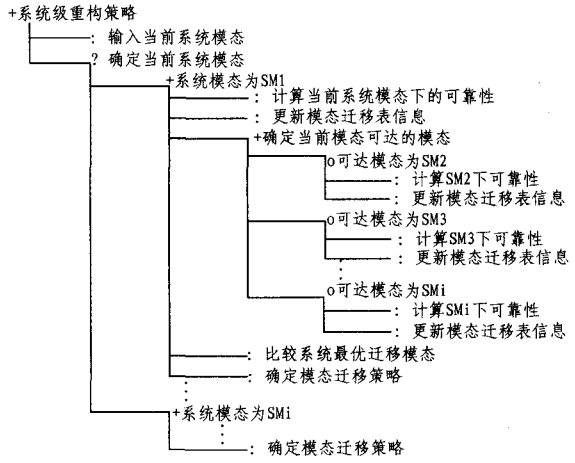

图 8 系统模态重构蓝图

在图 8 所示的重构蓝图中，输入当前系统模态之后，进入多选流程。若当前系统模态为 SM_1，则计算当前系统模态 SM_1 下的软件可靠性、安全性等属性，并在重构信息表中更新属性信息。接着判断当前系统模态 SM_1 可以迁移的模态。若可达模态 SM_i，则计算系统模态 SM_i 下的软件可靠性、安全性等属性，并在重构信息表中更新属性信息。当所有的可达模态计算完毕之后，比较系统最优的模态迁移，并确定模态迁移策略。按照图 8 所示，同样可以得到使命模态和功能模态的规划蓝图。系统按照规划蓝图，完成在系统模态层、使命模态层和功能模态层的动态重构。

结束语 本文基于软件系统架构的层次关系，研究了复杂嵌入式计算任务的运行模态表示方法，分析了嵌入式软件系统任务模态的迁移关系，提出了基于 AADL 软件体系结构的嵌入式软件模态划分方法，制定了系统动态重构蓝图，并设计了基于模态的嵌入式软件动态重构实施方法。基于软件架构的模态分析及其动态重构，提高了复杂嵌入式软件系统的可靠性、安全性和重用性。

参 考 文 献

[1] ARINC SPECIFICATION 653-2, Avionics Application Software Standard Interface[S]

[2] River W. VxWorks[EB/OL]. http://www.windriver.com/products/vxworks/, 2009

[3] Craveiro J. Integration of generic operating systems in partitioned architectures[D]. University of Lisbon, 2009

[4] 胡军. 构件化嵌入式软件设计的分析与验证[D]. 南京：南京大学，2005

[5] Perry D E. Software engineering and software architecture[C]// Feng Yu-lin, ed. Proceedings of the International Conference on Software: Theory and Practice. Beijing: Electronic Industry Press, 2000: 1-4

[6] Wang Geng, Zhou Xing-she, Dong Yun-wei. Studying on AADL-based Architecture Abstraction of Embedded Software[C]// The 8th IEEE International Conference on Embedded Computing. 2009

[7] Feiler P H, Gluch D P, Hudak J J. The Architecture Analysis & Design Language(AADL): An Introduction[M]. Carnegie Mellon University, 2006

[8] Rugina A-E, Kanoun K, Kaâniche M. A System Dependability Modeling Framework Using AADL and GSPNs[R]. LAAS-CNRS. 2007

[9] Zhang Chen-yu, Yang Zhi-yi, Dong Yun-wei. Research and Assessment of the Reliability of a Fault Tolerant Model Using AADL[R]. 2008 Advanced Software Engineering & Its Applications, 2008

[10] 刘建宾. 过程蓝图设计方法学[M]. 北京：科学出版社，2005