ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.9, September 2010, pp.2118−2134
doi: 10.3724/SP.J.1001.2010.03643

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

软件体系结构层次的软件适应性预测模型*

高 晖+, 张 莉, 李 琳

(北京航空航天大学 软件工程研究所,北京 100191)

Software Flexibility Prediction Model in Software Architecture Level

GAO Hui+, ZHANG Li, LI Lin

(Software Engineering Institute, BeiHang University, Beijing 100191, China)

+ Corresponding author: E-mail: gaohui@cse.buaa.edu.cn

Gao H, Zhang L, Li L. Software flexibility prediction model in software architecture level. *Journal of Software*, 2010,21(9):2118−2134. http://www.jos.org.cn/1000-9825/3643.htm

Abstract: In this paper, the prediction model based on Bayesian is constructed for presenting cause and effect relationships between software structure features and software flexibility by consulting the experiential data and expert knowledge. In addition, the Bayesian networks learning technique is extended to find out the weak causal relationships in the software flexibility prediction model constructing. Finally the method and examples using the prediction model to assess software flexibility in the software architecture level are presented.

Key words: software flexibility; software flexibility prediction; software architecture

摘 要: 结合经验数据和专家知识,基于贝叶斯网建立了软件体系结构层次的结构特征、变化原因与软件适应性之间的因果关联模型,即软件体系结构层次的软件适应性预测模型,扩展贝叶斯网学习算法解决了该预测模型中较弱因果关系的发现问题.最后给出应用预测模型在软件体系结构层次上评估软件适应性的方法和实例.

关键词: 软件适应性;软件适应性预测;软件体系结构

中图法分类号: TP311 **文献标识码**: A

软件适应性(software flexibility)是人们在研究软件适应变化能力时提出的术语.在 IEEE 软件工程术语定义标准[1]中,软件适应性定义为一个系统或构件为适应业务应用和环境的改变(不包括设计时能够确定的改变)进行变更的难易程度.Eden 等人更进一步地说明了软件适应性的特点[2].对于软件适应性的特点,一些研究者分别从软件设计技术和软件度量技术的研究角度进行说明,其中比较有影响的观点是[2]:(1) 不应该将软件适应性作为一个绝对的量来理解或者度量;(2) 虽然软件变化在实际发生前是不可知的,但软件变化的类型是有限的,所以在设计时可以根据软件变化的类型进行预测来提高软件的适应能力;(3) 要提高软件适应性就是要在特定变化发生时使某些结构方面的变化独立于其他方面以增加系统的鲁棒性.20 世纪 90 年代,软件体系结构[3,4]概念被明确提出,并在软件工程界获得了广泛关注.软件体系结构是软件开发过程早期的软件制品,是一个高度抽象的软件设计模型.软件体系结构通过合理的抽象屏蔽细节、突出关键问题的设计思想将发挥越来越重要的作用.通过对几十种在软件项目中实际使用的软件体系结构(软件体系结构模式)[5−7]的调查可以发现, 65%

* Supported by the National Natural Science Foundation of China under Grant Nos.60773155, 90818017 (国家自然科学基金)
Received 2008-07-29; Revised 2009-02-12; Accepted 2009-04-29

以上的软件体系结构模式对软件适应性产生影响.所以,在软件体系结构层次研究软件适应性是非常重要和必要的.

软件适应性研究的最终目标就是提高不同抽象层次的软件制品对外界影响因素的适应能力.要达到这个目标,一般来说可以从两个不同的研究思路出发:(1) 在设计过程采用适应性设计技术,比如面向对象设计、设计模式、面向方面设计、MDA 等技术[8−13],或者采用合理的开发过程[14]来提高软件的适应性;(2) 通过软件评估或测试来保证软件的适应性,从而降低软件在适应性方面的风险.在软件体系结构层次,可以利用很多基于场景的评估方法对软件的质量属性进行评估和验证,比如 SAAM,ATAM 方法[15]等都可以用于对软件适应性的评估.上述两种研究都需要解决一个基本问题,即哪些因素会影响到软件的适应性.通过经验数据建立软件适应性与其他因素之间的关系模型,才能从本质上了解哪些因素对软件适应性产生影响,从而可以进一步在软件开发周期中利用这些知识(关系模型)来指导设计或评估,最终提高软件适应性.对于影响软件适应性的因素,可以从以下方面进行考虑:(1) 软件本身的因素,实践人员和研究人员一致认为:在软件产品结构和质量之间可能存在着一定的联系[2];(2) 开发人员的因素,包括人的开发素质、对原有软件的理解,也可能影响软件适应性;(3) 其他因素,包括选择的编程语言、软件的开发过程、开发机构的组织结构都可能对软件适应性产生一定的影响.软件体系结构作为软件开发周期中非常重要的设计模型,描述了软件的基本结构特征,因此本文研究的重点就是在软件体系结构层次上研究软件结构与软件适应性之间的基本关系,为软件适应性设计、软件体系结构层次的适应性评估技术提供理论和实践的基础.

本文的主要贡献是:

(1) 建立软件体系结构层次上的软件适应性预测模型(SAFlexBBN).该模型利用贝叶斯网建立了软件体系结构层次的结构特征、软件变化原因与软件适应性之间的关系.该模型是经验数据构造知识的具体表现形式.与传统的度量和预测指示器方法相比,贝叶斯网表示了相关各个因素之间的因果关系,利用因果关系预测特定因素.传统的度量和预测方法是利用内部度量作为指示器来预测外部度量,而不是直接描述各个因素之间的关系.

(2) 研究了如何从 SAFlexBBN 获取经验知识,并进一步在软件体系结构评估中的应用.该研究给出了在 SAFlexBBN 的基础上获取到预测因子(即软件体系结构的结构和软件变化原因等信息)时,评价软件适应性的方法.

(3) 研究了贝叶斯理论在软件质量的评估和预测中的应用,为其在软件工程领域的应用提供了实践经验.贝叶斯网作为一种理论模型,在软件工程研究中具有较为广泛的应用领域.本文针对一些在应用贝叶斯网时可能出现的问题,如较弱因果关系的发现、经验数据与专家经验结合等,为相关的研究提供了可借鉴技术手段.

本文第 1 节分析和说明在软件体系结构层次上软件结构与软件适应性之间存在的关系是一种不确定性的因果关系.第 2 节给出基于贝叶斯网的软件适应性预测模型(SAFlexBBN),并对该预测模型的构造方法进行说明.第 3 节说明软件适应性预测模型的应用.最后给出相关研究以及对本文研究工作的总结.

1 软件结构与软件适应性之间的不确定性因果关系

为了利用理论模型对软件结构与软件适应性之间的关系进行建模,首先需要确定这两者之间关系的特点.通过分析,软件结构与软件适应性存在的关系是一种不确定的因果关系.

不确定性在软件的开发过程和软件产品中是固有的、不可避免的特征[16].在软件结构一定的情况下,最终软件产品的适应性可能是不一样的.但经过大量实践又可以发现,软件结构在一定程度上又影响着软件适应性,大量的软件模式可以说明这个问题.软件模式,包括设计模式和软件体系结构模式,是设计经验的总结,用于解决特定的问题.通过对几十种软件体系结构模式和设计模式的分析可以发现,大多数软件模式对软件适应性产生影响.结合实践经验,软件结构在某种程度上对软件的适应性产生了很大的影响.例如,在软件体系结构评估中,根据给出的软件体系结构模型就可以观察到模型的结构特征,比如结构复杂性、系统耦合度、系统形态等特征,对软件适应性进行预测.预测结果一般由概率给出.为了表示软件结构与软件适应性的这种不确定性关

系,只能选择一种能够处理不确定性的理论模型.

对于因果关系的判断,Babbie 等研究者已经提出了 3 个判别标准[17]:

(1) 原因在时间上或逻辑上发生在结果之前;

(2) 两个变量必须相互关联;

(3) 两个变量间的相关性不是伪造关系的结果.

软件结构在软件设计阶段就确定了,而软件适应性则是软件在维护阶段对变化的适应能力.所以从时间上看,软件结构的确定发生在软件适应性确定之前.这就满足了第 1 个标准.

从经验的角度出发,在软件结构的特征与软件适应性之间存在以下一些关系:软件结构越复杂可能使得软件适应性越差;软件耦合度越高可能使得软件适应性越差;系统形态也可能对软件适应性产生影响.根据这样一些经验的关系,可以初步说明软件结构与软件适应性之间是相互关联的,也可以初步说明两者的关系满足第 2 个标准.更进一步,需要一种理论的方法来证明和验证软件结构特征与软件适应性的相关关系.

第 3 个标准的验证是比较困难的,可能存在干扰变量、非直接原因等因素造成伪造关系.在相关性欺骗关系的检查中,如果只考虑变量之间两两关系是很难判断伪造关系的.但选用合适的理论模型就可以很好地解决这个问题,比如:本文所采用的贝叶斯网模型可以建立所有变量之间的关系,变量之间是否存在因果关系以及因果关系的程度都能够表示.通过贝叶斯网的学习,干扰变量、非直接原因导致的伪造关系在贝叶斯网中可以明显地表现出来.

所以,通过初步判断可以大致得到软件结构与软件适应性存在因果关系.

2 基于贝叶斯网的预测模型

根据前面的分析,软件结构特征和软件适应性之间表现为一种不确定性的因果关系,因此需要选择一种能够表示不确定性因果关系的数学模型来构造预测模型.另外,在选择数学模型时需要考虑构造模型和预测的方法.本文选择了贝叶斯网作为软件适应性预测模型的理论基础.

2.1 贝叶斯网在软件适应性预测中的应用分析

贝叶斯网理论主要用于解决在不确定环境中的知识表示、问题推理.根据前面的分析,软件结构特征和软件适应性之间也表现为一种不确定性的因果关系.另外,由于测试所得到的经验数据并不满足正态分布,而且可以判断经过离散化处理以后的经验数据满足多项式分布,所以本文中选择离散性的贝叶斯网来建立软件适应性预测模型.

更进一步地,在软件适应性的预测技术研究中,需要解决两个问题:(1) 如何利用经验数据构造软件适应性的预测模型;(2) 如何利用预测模型,在软件体系结构设计或评估中预测软件的适应性.贝叶斯网的学习和推理理论的研究为解决这两个问题提供了技术基础.下面就从贝叶斯网的特点出发,介绍贝叶斯网在软件适应性预测研究中的优点和缺点.

为了合理地应用贝叶斯网解决软件适应性预测中的问题,结合贝叶斯网在知识表示、推理以及贝叶斯网构造方面具有的优点进行如下应用分析:

(1) 贝叶斯网有效地将图论和概率理论有机的结合,不但具有了形式的概率理论基础,同时也具有更加直观的知识表示形式.在软件适应性预测技术中,利用贝叶斯网来描述软件结构和软件适应性的因果关系,不仅直观地表示软件结构特征与软件适应性的关系,而且可以提供丰富的含义.

(2) 贝叶斯网可以表达事件或属性间的带有不确性的相互关系,也可以在获取某些证据的情况下进行推理和预测.在软件适应性预测技术中,选择贝叶斯网作为理论基础,就可以在统一理论的基础上完成预测模型的定义、预测模型的构造以及预测模型的推理.

(3) 贝叶斯网没有确定的输入/输出节点,网络中任何节点都可以充当输入/输出.这一特点使得贝叶斯网具有其他模型没有的好处:可以处理不完整的输入数据集.在软件适应性预测中,由于在软件设计过程中信息是逐步获取的,或者由于软件设计模型描述得不完整,这就造成了信息的获取很可能是不完整的.所以,如何解决证

据不完整情况下的预测就非常重要了.特别是在软件体系结构层次上,预测模型的定义者往往考虑了很多因素,但在实际工作中,由于设计的抽象层次、描述方法等方面的差别,有些信息在预测时是不能获取的.贝叶斯网建立了各个事件或属性间的关系,任何事件都可以是证据或者是结果,所以具有处理不完整证据的能力.

(4) 结合贝叶斯方法来构造贝叶斯网可以有效地结合领域知识(先验概率)和观察数据.领域知识对于现实世界的分析是非常重要的,特别是当数据较少或昂贵时.在软件适应性预测模型的构造中,观察数据的获取是非常困难的,所以结合领域知识和能够获取的观察数据是非常重要的.

(5) 利用贝叶斯方法来构造贝叶斯网可以有效地避免数据的过渡拟合.因为利用贝叶斯方法,每个观察数据可以增量地降低或升高某假设的估计概率,而不是完全地接受或拒绝该假设,这提供了一种比其他算法更合理的学习途径.而其他算法则会在某个假设与任意观察数据不一致时完全去掉该假设.在软件适应性预测模型构造中,这一优点可以促进最大限度地利用观察数据.

以上分析说明了在软件适应性预测技术中应用贝叶斯网的一些优点.然而,贝叶斯网在应用过程中仍然存在一些难点和问题:

(1) 贝叶斯网的构造困难.贝叶斯网可以很好地结合先验知识和观察数据,但获取先验知识和足够的观察数据,在软件适应性预测模型的构造中,甚至在软件工程的相关研究中,都是一个非常困难的问题.在构造软件适应性预测模型时,首先需要确定可以利用的领域知识来帮助建立贝叶斯网的结构,这些领域知识包括两个部分:网络中节点的选取和按照时间或逻辑上先后顺序确定节点间因果关系的顺序.利用这些先验知识,可以增强预测模型的合理性和正确性.利用软件体系结构模式和设计模式来获取设计模型,可以保证数据具有一定的数量,降低了数据收集的难度.另外,由于模式是经验的总结,通过模式收集的数据是具有代表性的.

(2) 离散贝叶斯网引起的数据缺失问题.当连续的数据在离散化时,信息发生衰减,可能造成贝叶斯网推理、学习的准确性降低.在软件适应性预测技术中,由于各组观察数据并不满足某种分布,所以就需要在应用贝叶斯网进行训练前将数据进行离散化处理.离散化处理实际上是一种等级划分或分类处理,虽然信息的衰减是绝对的、不可避免的,但在相应的应用情况下将观察数据中的分类信息表示出来,就可以在某种程度上相对地减少信息衰减.在软件体系结构层次的软件适应性预测中,对度量数据进行离散化处理是合理的,采用合理的等级划分可以使预测模型的物理意义更加明显.

2.2 软件适应性预测的贝叶斯网

2.2.1 软件适应性和软件结构的度量

本文构造了一个用于表达软件体系结构层次的软件结构与软件适应性之间关系的贝叶斯网模型(SAFlexBBN),它是一个定量化的模型,因此首先需要将软件适应性和软件结构这两个高层的软件质量属性进行量化.因此,软件适应性和软件结构在软件体系结构层次的度量技术是本文的研究基础,本课题组对这两方面进行了一定的研究[18,19],由于不是本文重点,所以只是简要介绍其研究成果.

- 软件适应性度量

根据文献[18]中的研究,软件适应性很难用单一的软件质量子特征来表示.根据软件适应性与软件变化的关系,定义如下的质量子特征和度量来量化软件适应性,见表 1.

在软件适应性度量中,需要说明两点:

(1) 软件变化的影响范围是有区别的.软件体系结构由许多构件构成,而这些构件又属于不同的类别.当变化的构件范围只是某一类型的一个或某几个构件,而不是这一类型中所有构件都发生变化时,将这种变化称为 1 类变化.当变化的构件范围是某一种类型中的所有构件时,将这种变化称为 2 类变化.一般认为:如果软件只发生 1 类变化,其适应性是较好的;如果软件发生较多的 2 类变化,其适应性则较差.因此,在定义软件适应性的度量时,分别针对 1 类变化和 2 类变化来定义度量用于更好地表现软件适应性;

Table 1 Metric of software flexibility in software architecture level

表 1 软件体系结构层次的软件适应性的度量定义[18]

Quality feature	Quality metric	Metric definition
Causes of changes	The type of change cause	The causes of changes include: function requirement changes, quality requirement changes, technique mechanism changes.
Scale of changes	The max scale of the 1st class change	There is a Change Impacted Graph when software architecture changing, and the max scale of the 1st class change is defined $MaxChangeSize\ 1=\max(\{v_i.ChangeSize\ 1 \mid v_i \in CIG.V\_Changes\})$
	The mean scale of the 1st class change	There is a Change Impacted Graph when software architecture changing, and the mean scale of the 1st class change is defined $MeanChangeSize\ 1=\frac{\sum_{v_i \in CIG.V\_Chagnes} v_i.ChangeSize\ 1}{\mid CIG.V\_Chagnes \mid}$
	The max scale of the 2nd class change	There is a Change Impacted Graph when software architecture changing, and the max scale of the 2nd class change is defined $MaxChangeSize\ 2=\max(\{v_i.ChangeSize\ 2 \mid v_i \in CIG.V\_Changes\})$
	The mean scale of the 2nd class change	There is a Change Impacted Graph when software architecture changing, and the mean scale of the 2nd class change is defined $MeanChangeSize\ 2=\frac{\sum_{v_i \in CIG.V\_Chagnes} v_i.ChangeSize\ 2}{\mid CIG.V\_Chagnes \mid}$
Ratio of changes	The module change ratio	When software architecture changing, the module change ratio if defined $ChangeRatio=\frac{\text{修改的构件数+增加的构件数}}{\text{系统原有构件数+增加的构件数−删除的构件数}}$

(2) 软件元素之间存在依赖关系使得变化并不是孤立的.当某个元素发生变化时,可能会波及到软件的其他元素.在定义软件适应性度量时,考虑的并不是单个孤立的软件变化,而是利用了软件变化及变化之间的关系来定义软件适应性度量.因此,文献[18]定义了变化影响图对变化及其影响关系进行描述.变化影响图是一个以变化(软件变化)为节点、变化的影响关系为边的有向图.其中,变化为一个五元组,*ElementTemp* 为变化作用的软件元素,*ChangeSize* 为变化规模,*OpType* 为变化操作类型,*ImpactedChanges* 为该变化引发的其他变化集合,*Weight* 为变化加权系数.变化影响图是计算变化规模和变化比率度量的基础.图 1 给出了一个变化影响图的示例,该变化影响图给出了 MVC 模式中当 Model 发生变化时的一种变化影响图,其中,modModel 项表示修改属于 Model 类型的所有构件:0 表示发生 1 类变化的个数,1 表示发生了 2 类变化,MOD_COMP 表示修改构件.在 MVC 模式中,由于 Controller 和 View 都直接依赖于 Model,当 Model 发生变化时,Controller 和 View 也可能发生变化.更进一步,图中还给出了 Model,Controller,View 的变化规模和变化操作类型等信息,这些信息用于计算软件适应性相关的度量.

- 软件结构度量

文献[19]借鉴了传统的软件结构度量技术,在软件体系结构层次上定义了一组新的度量来测量软件体系结构模型的结构特征.表 2 给出了软件体系结构层次上的软件结构质量子特征和度量.

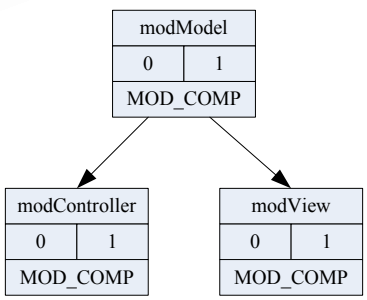


Fig.1 Example of change impacted graph

图 1 变化影响示意图[18]

Table 2 Metric of software structure in software architecture level

表 2 软件体系结构层次的软件结构的度量定义[19]

Quality feature	Quality metric	Metric definition
Complexity of software structure	The complexity of system roles CP_SYS_R	$Role(x)$ is the template role collection of the specific connector's type x in the software architecture model M, then the complexity of system roles is defined $CP\_SYS\_R = \frac{\sum_{x \in CN\_TYPE} (\mid Role(x) \mid)^2}{\mid CP\_TYPE \mid}$
	The complexity of system messages CP_SYS_OM	$Messages(x)$ is the output message collection of the specific connector's type x in the software architecture model M, then the complexity of system messages is defined $CP\_SYS\_OM = \frac{\sum_{x \in CN\_TYPE} (\mid Messages(x) \mid)}{\mid CP\_TYPE \mid}$
Software morph	The size of component's types MP_SYS_CPTSIZE	The size of component's types in the software architecture model M is defined $MP\_SYS\_CPTSIZE = \mid CP\_TYPE \mid$
	The size of connector's types MP_SYS_CNTSIZE	The size of connector's types in the software architecture model M is defined $MP\_SYS\_CNTSIZE = \mid CN\_TYPE \mid$
	The size of components MP_SYS_CPSIZE	The size of components in the software architecture model M is defined $MP\_SYS\_CPSIZE = \sum_{t \in CP\_TYPE} \mid CP(t) \mid$
	The size of connectors MP_SYS_CNSIZE	The size of connectors in the software architecture model M is defined $MP\_SYS\_CNSIZE = MP\_SYS\_CNTSIZE$
	The depth of system MP_SYS_DEPTH	Role(x) is the role collection of the specific connector's type x in the software architecture model M, then the depth of system is defined $MP\_SYS\_DEPTH = \max(\mid Role(x) \mid), x \in CN$
	The width of system MP_SYS_WIDTH	The width of system in the software architecture model M is defined $MP\_SYS\_WIDTH = \max(\mid CP(t) \mid), t \in CP\_TYPE$
	The connection density of system MP_SYS_CD	The connection density of system in the software architecture model M is defined $MP\_SYS\_CD = \frac{\sum_{x \in CP\_TYPE} (dependence\_count(x))}{\mid CP\_TYPE \mid}$
	The type tree's complexity of system MP_SYS_TIP	The type tree's complexity of system in the software architecture model M is defined $MP\_SYS\_TIP = \frac{\text{多于构件类型依赖图生成树的边数}}{\text{多于生成树的最大边数}}$
Reusability	The internal reusability RU_SYS_INTER	The internal reusability in the software architecture model M is defined $RU\_SYS\_INTER =$ 构件类型依赖图的边数－构件类型依赖图生成子树的边数
Coupling	The global coupling CL_SYS_MEDIAN	The global coupling in the software architecture model M is defined $CL\_SYS\_MEDIAN = median(\{CL\_CP\_IND(x,y)\})$ And the function median is count the mean value of elements in collections. $CL\_CP\_IND(x,y) = i + \frac{n}{n+1}, x \in CP\_TYPE, y \in CP\_TYPE$

2.2.2 SAFlexBBN 的结构和参数

SAFlexBBN 是一个贝叶斯网,该模型由两部分组成,即贝叶斯网的结构和参数.贝叶斯网的结构是一个有向无环图,图的节点为第 2.2.1 节中定义的软件适应性度量和软件结构度量.SAFlexBBN 的结构如图 2 所示,该结构定性地给出了各个节点之间的因果关系,即两个节点间如果存在有向边则表示两者之间存在因果关系,否则两者独立;SAFlexBBN 的参数如图 3 所示,参数就是每个节点的条件概率表.条件概率表以条件概率的形式给出了存在有向边的节点之间量化的因果关系.

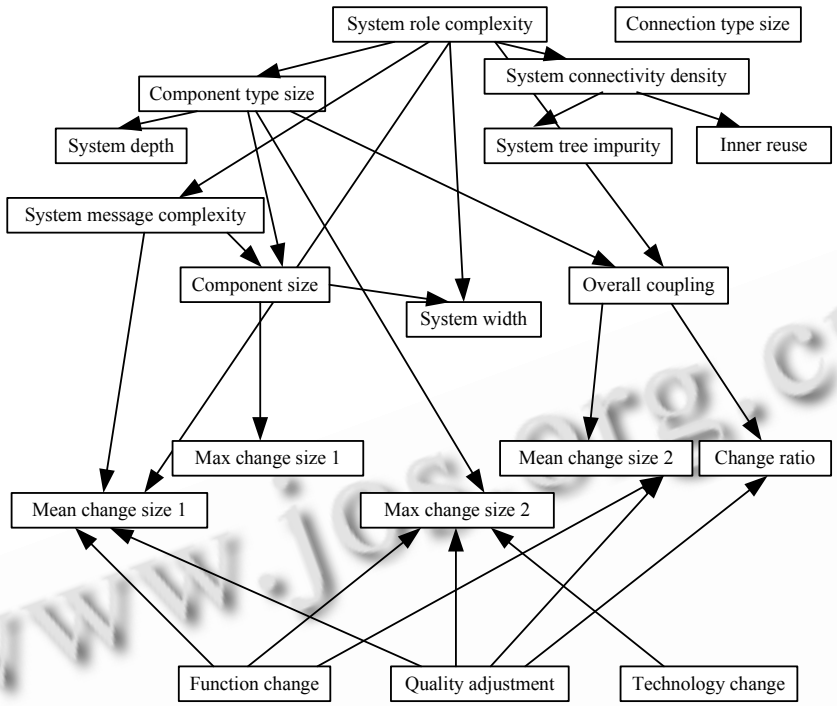


Fig.2 SAFlexBBN's structure

图 2 SAFlexBBN 的结构

从 SAFlexBBN 的结构可以看出,节点分为 3 大类:与软件结构特征相关的 11 个节点;与软件变化原因相关的 3 个节点;与软件变化规模和变化比率度量相关的 5 个节点.这 3 类节点之间存在如下几种因果关系:(1) 软件结构特征相关的节点是软件变化规模和变化比率的原因;(2) 软件变化原因相关的节点是软件变化规模和变化比率的原因;(3) 软件结构特征相关的节点之间存在因果关系.

SAFlexBBN 的参数首先规定了每个节点的取值.在本文中为每个节点定义了离散型的等级值,而不是直接利用连续型的度量值.每个节点的等级值与其度量值是相关的.在建立等级值和度量值的关系时,利用了聚类的方法对经验数据进行分析,并结合软件适应性和软件结构度量的实际含义给出了对应关系,见表 3.另外,图 3 中也分别给出了 SAFlexBBN 各个节点的条件概率表,其中,L 表示低,M 表示中,H 表示高,Z 表示 0,Y 表示是,N 表示否.

2.3 SAFlexBBN的构造方法说明

贝叶斯网的学习理论为 SAFlexBBN 的构造提供了基本的理论基础.本节将对在 SAFlexBBN 的构造方法进行简要的说明.

构造 SAFlexBBN,即确定 SAFlexBBN 的结构和参数,通常有两种方法:

(1) 通过领域专家人工建立 SAFlexBBN;

(2) 利用贝叶斯网的学习理论构造 SAFlexBBN.

本文采用了第(2)种方法.在软件工程领域,依靠贝叶斯网的学习理论来构造贝叶斯网在相关文献中还没有可以借鉴的经验和成果.根据贝叶斯网学习的一般过程,预测模型构造的主要步骤如图 4 所示.

Table 3 Metric level

表 3 度量的等级划分表

Metric	Level	Range	Metric	Level	Range
The max scale of the 1st class change	High	(5,+∞)	The size of components	High	(100,+∞)
	Low	[0,5]		Middle	(50,100]
The mean scale of the 1st class change	High	(3,+∞)		Low	(0,50]
	Low	[0,3]	The size of connectors	High	(3,+∞)
The max scale of the 2nd class change	High	(3,+∞)		Low	(0,3]
	Low	(0,3]	The depth of system	High	(6,+∞)
	Zero	0		Low	(0,6]
The mean scale of the 2nd class change	High	(2.5,+∞)	The width of system	High	(80,+∞)
	Low	(0,2.5]		Middle	(30,80]
	Zero	0		Low	(0,30]
The module change ratio	High	(0.6,1]	The connection density of system	High	(1.5,+∞)
	Middle	(0.3,0.6]		Low	(0,1.5]
	Low	[0,0.3]	The type tree's complexity of system	High	(0.4,+∞)
The complexity of system roles	High	(8,+∞)		Low	(0,0.4]
	Low	(0,8]	The internal reusability	High	(4,+∞)
The complexity of system messages	High	(8,+∞)		Low	(0,4]
	Low	(0,8]	The global coupling	High	(3.5,+∞)
The size of component's types	High	(10,+∞)		Middle	(2,3.5]
	Middle	(6,10]		Low	(0,2]
	Low	(0,6]	The size of connector's types	High	(3,+∞)
				Low	(0,3]

System role complexity		
L	0.87	
H	0.13	

Sysem role complexity		L	H
Component type size	L	0.85	0.76
	M	0.01	0.12
	H	0.05	0.12

Component type size		L	M	H
System depth	L	0.97	0.23	0.31
	H	0.03	0.77	0.69

System message complexity		L			H		
Component type size		L	M	H	L	M	H
Component size	L	0.96	0.7	0.21	0.33	0.33	0.33
	M	0.02	0.15	0.21	0.55	0.33	0.33
	H	0.02	0.15	0.58	0.12	0.34	0.34

System role complexity		L	H
System connectivity density	L	0.93	0.61
	H	0.07	0.39

Function change		
Y	0.61	
N	0.39	

System role complexity		L	H
System message complexity	L	0.93	0.39
	H	0.07	0.61

System role complexity		L			H		
Component size		L	M	H	L	M	H
System width	L	0.96	0.58	0.21	0.7	0.21	0.33
	M	0.02	0.21	0.21	0.15	0.58	0.33
	H	0.02	0.21	0.58	0.15	0.21	0.34

Quality adjustment		
Y	0.56	
N	0.44	

Technology change		
Y	0.19	
N	0.81	

System connectivity density		L	H
System tree impurity	L	0.93	0.5
	H	0.07	0.5

Component size		L	M	H
Max change size 1	L	0.96	0.59	0.69
	H	0.04	0.41	0.31

System connectivity density		L	H
Inner reuse	L	0.97	0.23
	H	0.03	0.77

Overall coupling		L				H			
Function change		N		Y		N		Y	
Quality adjustment		N	Y	N	Y	N	Y	N	Y
Mean change size 2	Z	0.17	0.09	0.71	0.18	0.24	0.16	0.65	0.25
	L	0.57	0.6	0.17	0.52	0.52	0.75	0.26	0.68
	H	0.26	0.31	0.12	0.3	0.24	0.09	0.09	0.07

System role complexity		L			H		
Component type size		L	M	H	L	M	H
Overall coupling	L	0.46	0.77	0.69	0.18	0.5	0.5
	H	0.54	0.23	0.31	0.82	0.5	0.5

Overall coupling		L		H	
Quality adjustment		N	Y	N	Y
Change ratio	L	0.58	0.48	0.64	0.25
	M	0.36	0.44	0.28	0.68
	H	0.06	0.08	0.08	0.07

System role complexity		L								H							
System message complexity		L				H				L				H			
Function change		N			Y			N		Y		N		Y			
Quality adjustment		N		Y		N		Y		N		Y		N		Y	
Mean change size 1	L	0.79	0.91	0.88	0.89	0.5	0.5	0.5	0.58	0.5	0.5	0.58	0.64	0.5	0.64	0.64	0.64
	H	0.21	0.09	0.12	0.11	0.5	0.5	0.5	0.42	0.5	0.5	0.42	0.36	0.5	0.36	0.36	0.36

Component type size		L								M								H							
Function change		N				Y				N				Y				N				Y			
Quality adjustment		N		Y		N		Y		N		Y		N		Y		N		Y		N		Y	
Technology change		N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y
Max change size 2	Z	0.33	017	0.11	0.19	0.72	0.52	0.22	0.33	0.33	0.28	0.24	0.28	0.38	0.33	0.33	0.33	0.33	0.28	0.33	0.33	0.44	0.33	0.28	0.33
	L	0.33	0.57	0.64	0.52	0.22	0.24	0.56	0.33	0.33	0.44	0.38	0.28	0.24	0.33	0.33	0.33	0.33	0.44	0.33	0.33	0.28	0.33	0.28	0.33
	H	0.34	0.26	0.25	0.29	0.06	0.24	0.22	0.34	0.34	0.28	0.38	0.44	0.38	0.34	0.34	0.34	0.34	0.28	0.34	0.34	0.28	0.34	0.28	0.34

Fig.3 SAFlexBBN's CPT

图 3 SAFlexBBN 的条件概率表

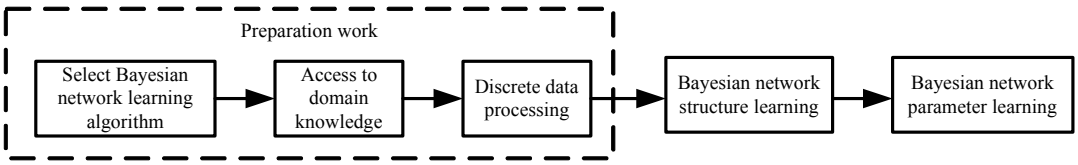


Fig.4 Steps of constructing SAFlexBBN

图 4 SAFlexBBN 构造的主要步骤

通过研究其他领域构造贝叶斯网的经验,当确定贝叶斯网结构时,在节点选择和先验知识的确定等方面需要一定的人工干预.另外,由于在软件工程领域的数据离散化处理和领域知识获取方面缺乏有效的理论和实践支持,所以本文在构造预测模型时多次迭代地执行数据离散化处理、领域知识获取、贝叶斯网结构和参数学习步骤,最终达到一个合理的结果.下面分别对构造过程中的一些主要问题进行说明.

- 领域知识获取

经验数据较少的情况下,软件适应性与软件结构之间关系的领域知识(先验知识)很大程度上影响了训练结果.在预测模型的构造中,贝叶斯网的结构实质上表现的是一种定性的结论,所以通过对其他研究成果的分析是比较容易转化为领域知识的.通过分析,本文在进行贝叶斯网构造学习时确定了如下的领域知识:

领域知识 1. 根据节点的发生时间或逻辑关系来初步划分原因节点和结果节点.在软件体系结构设计阶段,可以从软件体系结构模型中得到结构特征.软件的变化一般是发生在软件生命周期的后期.所以从发生的时间或者逻辑关系上,结构特征可能是软件适应性的原因.另外,在软件适应性度量中,软件变化是由于软件的问题域变化造成的,所以软件变化原因与软件结构是无关的,而软件变化原因可能是软件适应性的原因.所以,本文得到第 1 个先验知识,预测模型中的各个节点分为 3 大类:与软件结构特征相关的 11 个节点;与软件变化原因相关的 3 个节点;与软件变化规模和变化比率度量相关的 5 个节点.其中,前两类是原因节点,最后一类是结果节点.

领域知识 2. 一般情况下,预测模型中的各个节点之间的因果关系需由学习算法来确定,但根据领域知识或常识可以对一些不存在因果关系的节点进行判断.在软件适应性预测模型中,可以很容易地发现以下几种节点间不会存在因果关系:(1) 变化原因相关的 3 个节点之间不会存在因果关系;(2) 变化规模和变化比率相关的 5 个节点之间不会存在因果关系.

另外,需要特别指出的是,在运用贝叶斯网络的结构学习理论时(即采用扩展的 K2 算法时,详见本节中“SAFlexBBN 的结构学习方法”),需要对各个节点进行排序,这种排序实际上是对这些节点建立一种假设(即在后面的节点不会成为前面节点因素的原因节点).本文对软件结构特征进行了如下的排序:〈系统角色复杂度,系统消息复杂度,构件类型规模,连接件类型规模,构件规模,系统深度,系统宽度,系统连接性密度,系统类型树复杂度,内部重用度,全局耦合度,最大 1 类变化规模,平均 1 类变化规模,最大 2 类变化规模,平均 2 类变化规模,模块变化率〉,因此 SAFlexBBN 也就是在这种假设下得到的模型.

- 经验数据的获取

如何获得足够的训练数据是贝叶斯网在应用中遇到的一个非常棘手的问题,特别在软件工程领域,获得大量实际项目的体系结构模型是很困难的.而过去在软件工程领域应用贝叶斯网时,由于无法获取足够的数据,都是依靠专家的经验手工构造贝叶斯网(甚至是条件概率表).本文利用软件体系结构模式代替实际项目作为数据来源,减少了数据获取的难度,并且由于模式本身所具有的代表性,所以数据的质量也可以得到保证.基于模式的设计方法被大量应用于实际项目中,本文将应用模式的实际项目及文献[5−8]中描述的软件体系结构模式和设计模式对应的应用实例作为构造经验设计模型的来源.在建立 SAFlexBBN 时,本文从文献[5−8]共收集了 21 种软件体系结构模式和 7 种设计模式作为经验设计模型,并针对每种模式给出了 3 种变化假设,然后,通过软件适应性度量和软件结构度量,总共获得 84 组经验数据.其中,表 4 为部分变化场景集,表 5 为部分软件体系结构适应性度量数据,表 6 为部分软件体系结构模式结构度量数据.限于篇幅,完整的经验数据在文中省略.

Table 4 Set of change scenario

表 4 变化场景集

Pattern name	ID	Change scenario	The type of change cause	Active changes
Layer	S1_C1	…	…	…
	S1_C2	The dynamic function configuration is implemented by the framework manager to manage the components	function requirement change/quality requirement changes	*addPluginManager*=⟨*PluginManager*,1,0,*ADD_COMP*,*NULL*⟩ *modCol*=⟨*Collection*,0,1,*MOD_COMP*,*NULL*⟩ *modEntity*=⟨*Entity*,0,1,*MOD_COMP*,*NULL*⟩ *modUICtrl*=⟨*UIControl*,0,1,*MOD_COMP*,*NULL*⟩ *modWSCtrl*=⟨*WSControl*,0,1,*MOD_COMP*,*NULL*⟩ *modUI*=⟨*UI*,0,1,*MOD_COMP*,*NULL*⟩ *modMessageRecver*=⟨*MessageRecver*,0,1,*MOD_COMP*,*NULL*⟩
	S1_C3	…	…	…
PipeFiliter	S2_C1	…	…	…
…	….	…	…	…

Table 5 Flexibility metric of software architecture

表 5 软件体系结构适应性度量数据

ID	Causes of changes	Scale of changes				Ratio of changes
	The type of change cause	The max scale of the 1st class change	The mean scale of the 1st class Change	The max scale of the 2nd class change	The mean scale of the 2nd class change	The module change ratio
S1_C1	1	5	2.71	0	0	0.025
S1_C2	1,2	1	0.14	4	2.29	0.491
S1_C3	3	0	0	3	2.14	0.492
…	…	…	…	…	…	…

Table 6 Structure metric of software architecture

表 6 软件体系结构模式结构度量数据

Pattern ID	Pattern name	Complexity of software structure		Software morph								Reusabi-lity	Coupling
		CP_SYS_R	CP_SYS_OM	MP_SYS_CPTSIZE	MP_SYS_CNTSIZE	MP_SYS_CPSIZE	MP_SYS_CNSIZE	MP_SYS_DEPTH	MP_SYS_WIDTH	MP_SYS_CD	MP_SYS_TIP	RU_SYS_INTER	CL_SYS_MEDIAN
S1	Layer	7.54	7.31	13	2	366	2	7	123	0.92	0.0625	4	1.06
S2	Pipe-Filter	7	3.14	7	1	7	1	7	1	0.86	0	0	1
S3	Black-board	5	6.8	5	1	5	1	5	1	1.4	0.19	3	2.45
…	…	…	…	…	…	…	…	…	…	…	…	…	…

- SAFlexBBN 的结构学习方法

针对离散型的贝叶斯网,最常用的学习方法是Cooper和Herskovits在文献[20]中提出的方法.然而直接利用该学习算法构造贝叶斯网时,除了按照节点顺序确定领域知识外,不能处理其他的领域知识.另外,在经验数据较少的情况下,Cooper 和 Herskovits 的学习方法在一些较强因果关系存在的情况下,有可能忽略一些较弱的因果关系.在本文中主要体现在变化原因对于变化规模和变化比率的因果关系较强,而结构特征与这两者的因果关系相对较弱.

为了解决这两个问题,本文首先扩展了 Cooper 和 Herskovits 方法中的 K2 搜索算法,如图 5 所示.该算法可以支持以下两类领域知识:(1) 确定某些节点间存在因果关系;(2) 限制某些节点间存在因果关系.

图 5 中,函数 g 是 K2 算法中的搜索评价函数,计算方法见文献[21].函数 $Pred(i)$表示可能作为 i 节点的父节点的集合,即在有序节点集合 *nodes* 中 $index>i$ 的节点的集合.

```
//Function name: Ext_K2_Search
//Description: extended K2 search algorithm
//input parameters
//       nodes: sets of ordered nodes
//       permitRelation: causal relationship with permitted priori knowledge
//       forbidRelation: causal relationship with forbidden priori knowledge
//       data: the set of trained data
//       u: the number of maximum parent nodes
Ext_K2_Search(nodes,permitRelation,forbidRelation,data,u) {
     for (i=1<nodes.size()) {
          PAi.clear();
          //   add into the set of parent nodes directly if it is permitRelation.
          foreach z=Pred(i) {
               if ⟨i,z⟩ is in permitRelation) {
                    PAi.add(z);
               }
          }
          pOld=g(i,PAi,data);
          OKToProceed=true;
          while (OKToProceed && PAi.size()<u) {
               //find out the node which makes the biggest g in the set of probable parent nodes of node i
               z=findMax_g(i,PAi,Pred(i),data);
               //   quit the while loop if it is forbidRelation
               if (⟨i,z⟩ is in forbidRelation) {
                    break;
               }
               //   process the nodes that have been added into the set of parent nodes
               if (⟨i,z⟩ is in Pred(i)) {
                    contintue;
               }
                pNew=g(i,PAi.add(z),data);
                if (pNew>pOld) {
                     pOld=pNew;
                }
                else {
                     OKToProceed=false;
                     PAi.remove(z);
                }
          }
          output ("Node", i, "\nParents of this Node are:" PAi);
     }
}
```

Fig.5 Extend K2 searching algorithm

图 5 扩展的 K2 搜索算法

下面通过贝叶斯网学习理论来说明扩展后的 K2 搜索算法的正确性.在 Cooper 和 Herskovits 的方法中,假设了所有贝叶斯网结构的 $p(B_S)$都相等.但当通过先验知识对贝叶斯网的结构进行限制时,这些先验知识将贝叶斯网结构的样本分为两个部分:一部分是符合先验知识的结构;另一部分是不符合先验知识的结构.设符合先验知识的结构集合为 BS.式(1)给出了在给定先验知识的情况下对 $p(B_S)$的假设.

$$p(B_S)=\begin{cases}\dfrac{1}{|BS|}, & B_S\in BS\\ 0, & B_S\notin BS\end{cases} \tag{1}$$

在先验知识的情况下的联合概率计算公式,见公式(2).

$$p(B_S,D)==\begin{cases}p(B_S)\left(\prod_{i=1}^{n}\prod_{j=1}^{q_i}\dfrac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!\right), & B_S\in BS\\ 0, & B_S\notin BS\end{cases} \tag{2}$$

由于模型结构不符合先验知识时其联合概率为 0,所以只需要在满足先验知识的结构集合中进行搜索,模型选择准则见公式(3):

$$\max_{B_S}[p(B_S,D)]=\max_{B_S}[p(B_S)p(D/B_S)]=\max_{B_S\in BS}[p(B_S)p(D/B_S)]=c'\max_{B_S\in BS}[p(D/B_S)] \quad (3)$$

综上所述,对 K2 搜索算法的扩展,实质上只是将搜索空间限制在符合先验知识的结构集合中.所以,扩展后的 K2 搜索算法在符合公式(3)的先验概率的情况下是正确的.

在扩展的K2搜索算法的支持下,为了发现较弱的因果关系,软件适应性预测模型的结构学习可以分解为两个步骤:

第1步.在不包含变化原因的情况下,利用扩展的K2搜索算法得到一个预测模型的子模型,该子模型只包含与软件结构特征相关的 11 个节点、与软件变化规模和变化比率度量相关的 5 个节点;

第2步.将第1步中得到的子模型作为先验知识,再次利用扩展的K2搜索算法即可得到图2中SAFlexBBN的结构模型.

3 应用研究

SAFlexBBN 以条件概率的形式给出了各个节点间的关系.在软件体系结构设计和评估中,可以通过对软件体系结构模型的度量得到一些结构特征,然后利用 SAFlexBBN 就可以推测出软件变化规模和变化率取得每一种可能值的概率.通过 SAFlexBBN 计算未知的概率就是贝叶斯的推理.贝叶斯网推理包含确切的推理和近似的推理两类方法,而且已经有很多工具可以支持贝叶斯网的推理,比如 Microsoft 研究院的 MSBNx[21,22]和 KSU 的 BNJ[23].在本文的工作中,选择了贝叶斯网推理工具 MSBNx 来实现在软件体系结构设计层次上预测软件适应性.MSBNx 采用了一种 clique-tree 的遗传算法来实现确切的推理算法.

3.1 应用实例1利用预测模型来获取经验知识

下面通过 SAFlexBBN 的推理来验证一些经验知识.在 SAFlexBBN 中包含了很多经验知识,这些知识反映了经验数据中实际存在的关系.在软件适应性的研究中,软件的哪些特性会影响到软件的适应性是一个重要的基础问题.从 SAFlexBBN 中,本文可以得到以下一些经验性的结论.

- 不同的软件变化原因是影响软件变化规模和变化比率的关键因素.

利用软件适应性的预测模型,本文将软件变化原因作为观察值分别给出进行实验,计算结果见表 4.表中计算结果的单元格中使用 *x*/*y*/*z* 分别表示 3 次实验的计算结果.表 7 的计算结果进一步表明了软件变化原因与变化规模之间的关系.

Table 7 Result of experiment

表 7 实验计算结果

Node	Result		
The max scale of the 1st class change	Low 0.90/0.90/0.90	Middle —/—/—	High 0.10/0.10/0.10
The mean scale of the 1st class change	Low 0.82/0.84/0.73	Middle —/—/—	High 0.18/0.16/0.27
The max scale of the 2nd class change	Zero 0.67/0.14/0.19	Low 0.23/0.59/0.55	High 0.10/0.27/0.26
The mean scale of the 2nd class change	Zero 0.68/0.13/0.20	Low 0.22/0.68/0.52	High 0.10/0.19/0.28
The module change ratio	Low 0.61/0.36/0.61	Middle 0.32/0.58/0.31	High 0.07/0.07/0.07

实验 1.1. 当发生功能需求变更时,测试软件变化规模和模块变化率的分布.

实验 1.2. 当发生质量调整时,测试软件变化规模和模块变化率的分布.

实验 1.3. 当发生技术变化时,测试软件变化规模和模块变化率的分布.

- 软件的结构特征是影响软件变化规模和变化比率的因素.

进行下面一组实验可以用来说明一些软件结构对软件变化规模和变化比率的影响.从实验结果看,软件结

构特征虽然对软件变化规模和变化比率有一定的影响,但这种影响比软件变化原因的作用要弱.

实验 2.1. 测试系统角色复杂度对平均 1 类变化规模的影响.

在系统角色复杂性取值为高和低时,分别通过贝叶斯网推理计算平均 1 类变化规模的分布,见表 8.通过上面的结果可以看出,当软件发生变化时,如果系统角色复杂度越低,平均 1 类变化规模为低的概率就越大.

Table 8 Result of experiment

表 8 实验计算结果

Evidence node		The mean scale of the 1st class change	
		High	Low
The complexity of system roles	High	0.40	0.60
	Low	0.15	0.85

实验 2.2. 在发生质量调整的情况下,测试系统构件类型规模对最大 2 类变化规模的影响.

在质量调整的情况下,在构件类型规模取值为高、中和低时,分别通过贝叶斯网推理计算最大 2 类变化规模的分布,见表 9.通过上面的结果可以看出,当软件发生质量调整的变化时,如果构件类型规模越小,最大 2 类变化规模为低的概率就越大.

Table 9 Result of experiment

表 9 实验计算结果

Evidence node		The max scale of the 2nd class change		
		High	Low	Zero
The size of component's types	High	0.33	0.33	0.34
	Middle	0.24	0.38	0.38
	Low	0.11	0.64	0.25

实验 2.3. 在发生质量调整的情况下,测试全局耦合度对平均 2 类变化规模的影响.

在质量调整的情况下,在全局耦合度取值为高和低时,分别通过贝叶斯网推理计算平均 2 类变化规模的分布,见表 10.通过上面的结果可以看出,全局耦合度和平均 2 类变化规模未表现出明显的关系.

Table 10 Result of experiment

表 10 实验计算结果

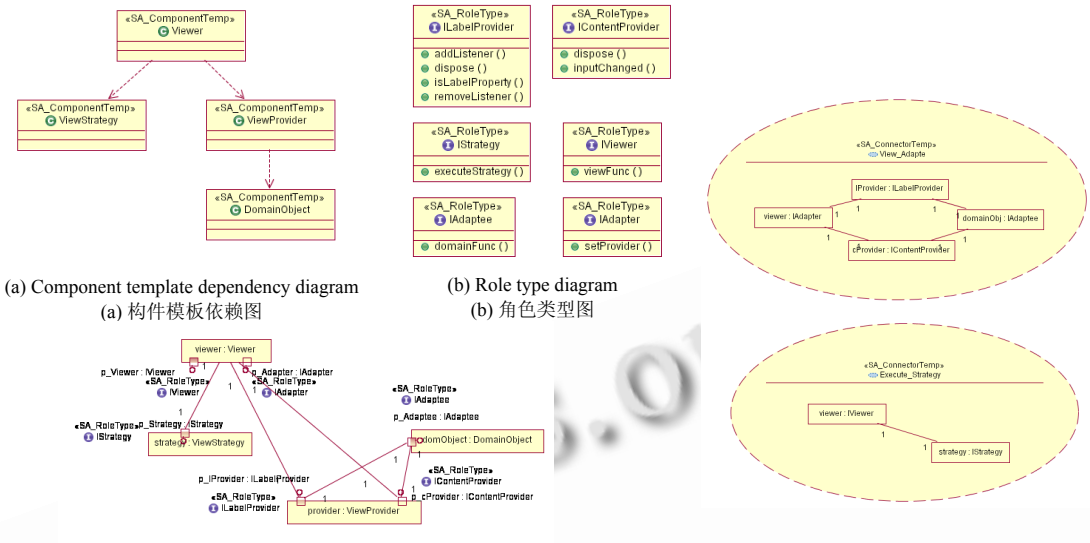
Evidence node		The mean scale of the 2nd class change		
		High	Low	Zero
The global coupling	High	0.09	0.75	0.16
	Low	0.31	0.60	0.09

综上所述,以上的各个测试通过观察预测模型的结构来建立一些经验知识,一般只考虑父子节点之间的关系,涉及到的节点数很少.利用这些测试的结果,就可以将预测模型中形式化的知识转化为人们更容易接受的一些规则.这些规则可以为设计或评估提供一些简单、直观的准则.

3.2 应用实例2在软件体系结构评估时对软件适应性进行预测

在进行软件体系结构评估时,可以在得到软件体系结构模型的基础上对软件的结构特征进行度量,然后利用软件适应性预测模型对该软件体系结构的适应性进行评估.JFace[24]是 Eclipse 体系中提供高层次应用支持的用户界面框架.本文就以 JFace 为例对其软件适应性进行评估.

首先,利用文献[25]中定义的软件体系结构建模语言 TADL 对 JFace 的软件体系结构进行建模,如图 6 所示.然后利用本文工作中实现的结构度量工具 StructureMetrics 对 JFace 的软件体系结构进行结构度量,见表 11.



(a) Component template dependency diagram
(a) 构件模板依赖图

(b) Role type diagram
(b) 角色类型图

(c) Structure diagram
(c) 静态结构图

(d) Connector diagram
(d) 连接件图

Fig.6 Architecture model of JFace
图 6 JFace 的软件体系结构模型

Table 11 Structure metric of JFace
表 11 JFace 的结构度量结果

Quality feature	Metric	Value	Level
Complexity of software structure	The complexity of system roles	5.0	Low
	The complexity of system messages	4.5	Low
Software morph	The size of component's types	4	Low
	The size of connector's types	2	Low
	The size of components	122	High
	The size of connectors	2	Low
	The depth of system	4	Low
	The width of system	58	Middle
	The connection density of system	0.75	Low
	The type tree's complexity of system	0.0	Low
Reusability	The internal reusability	0.0	Low
Coupling	The global coupling	1.81	Low

根据以上对软件体系结构的结构特征的度量结果,利用预测模型就可以在 MSBNx 工具中进行推理计算,得到以下预测值.表 12 中分别给出了在功能需求变更、质量调整和技术变化的情况下,变化规模和变化比率的预测值.

根据表 9 中的预测值,可以对 JFace 的软件适应性进行说明.

(1) 在进行功能需求变更的情况下,JFace 的基本设计思路发生变化的可能性较小;

(2) 在质量调整和技术变化的情况下,JFace 的修改难度会增加,但影响基本设计思路的可能性也较小;

(3) 综上所述,在各种变化原因下,由于 JFace 软件适应性都是较好的,从预测中可以知道各种软件适应性取值为高的概率都是较小的.

Table12 SAFlexBBN prediction result of JFace

表 12 JFace 的贝叶斯预测结果

Causes of changes	Scale and ratio of changes	Prediction result		
Function requirement changes	The max scale of the 1st class change	Low	High	–
	The mean scale of the 1st class change	0.69	0.31	–
	The max scale of the 2nd class change	Low	High	–
	The mean scale of the 2nd class change	0.88	0.12	–
	The module change ratio	Zero	Low	High
	The max scale of the 1st class change	0.72	0.22	0.06
	The mean scale of the 1st class change	Zero	Low	High
	The max scale of the 2nd class change	0.71	0.17	0.12
	The mean scale of the 2nd class change	Low	Middle	High
		0.58	0.36	0.06
Quality requirement changes	The max scale of the 1st class change	Low	High	–
	The mean scale of the 1st class change	0.69	0.31	–
	The max scale of the 2nd class change	Low	High	–
	The mean scale of the 2nd class change	0.91	0.09	–
	The module change ratio	Zero	Low	High
	The max scale of the 1st class change	0.11	0.64	0.25
	The mean scale of the 1st class change	Zero	Low	High
	The max scale of the 2nd class change	0.09	0.6	0.31
	The mean scale of the 2nd class change	Low	Middle	High
		0.48	0.44	0.08
Technique mechanism changes	The max scale of the 1st class change	Low	High	–
	The mean scale of the 1st class change	0.69	0.31	–
	The max scale of the 2nd class change	Low	High	–
	The mean scale of the 2nd class change	0.79	0.21	–
	The module change ratio	Zero	Low	High
	The max scale of the 1st class change	0.17	0.57	0.26
	The mean scale of the 1st class change	Zero	Low	High
	The max scale of the 2nd class change	0.15	0.52	0.33
	The mean scale of the 2nd class change	Low	Middle	High
		0.58	0.36	0.06

4 相关工作与展望

在软件工程领域,Ziv 最早阐述了在软件领域存在的不确定性,并利用贝叶斯网来描述这种不确定性.Ziv 研究了软件开发中各种软件制品之间的关系,比如软件需求、测试计划和软件模块之间的不确定关系[16],根据专家的经验就可以为软件项目建立描述软件制品关系的贝叶斯网,利用这个网络就可以对测试、维护性进行预测和决策.

Fenton 利用贝叶斯网代替传统的回归模型解决缺陷预测问题.Fenton 从理论上比较了因果关系模型(可以贝叶斯网表示)与传统预测模型之间的优劣,说明了因果关系模型在软件度量和预测技术研究中的作用[26,27].Fenton 的研究对贝叶斯网在软件度量和预测方面的应用提供了一个研究范例.

软件体系结构层次可靠性风险分析方法(ALRRA)[28]与本文在研究的路线上具有类似的地方,都是通过在软件体系结构层次研究特定软件质量属性的度量,并建立预测模型.区别在于研究的质量属性不同:ALRRA 研究的是可靠性,本文研究的是软件适应性;另外一个区别是建立预测模型的方法不同:ALRRA 利用了回归模型、神经网络等方式建立预测模型,本文则采用了贝叶斯网来建立预测模型.

基于贝叶斯网的软件体系评估方法(SAABNet)[29,30]与本文都采用了贝叶斯网作为理论模型.但 SAABNet 通过人工来建立贝叶斯网,本文则是结合人的经验作为先验知识,并收集经验数据,通过贝叶斯网学习技术自动构造贝叶斯网.人工构造贝叶斯网在确定参数时有很大的困难,而且不利于贝叶斯网的完善.

本文工作探索了软件体系结构层次的软件适应性预测技术,下面一些工作值得进一步研究:

(1) 软件适应性度量和预测技术在软件设计和评估中的应用研究.Internet 的出现对软件提出了新的要求,

也对软件适应性提出了更高的要求.研究基于Internet软件的适应性是一个重要的发展方向.本文的成果为研究该类软件的适应性提供了基础方法,可以用于研究基于 Internet 软件的适应性.另外,在应用研究的同时,根据贝叶斯网理论可以在现有的预测模型下增加经验数据来修正预测模型.

(2) 预测因子的扩展研究.本文将软件结构作为预测因子来研究软件适应性的预测模型,但开发人员、开发过程、软件的其他特征也可能成为软件适应性的预测因子.在经验数据能够得到的情况下,研究其他的预测因子可以补充现有的预测模型.

References:

[1] IEEE Standards Board. IEEE standard glossary of software engineering terminology 610.12-1990. Los Vol.1. Los Alamitos: IEEE Press, 1999.

[2] Eden AH, Mens T. Measuring software flexibility. IEE Proc. Software, 2006,153(3):113−125.

[3] Perry DE, Wolf AL. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 1992,17(4): 40−52. [doi: 10.1145/141874.141884]

[4] Garlan D, Shaw M. An introduction to software architecture. In: Ambriola V, Tortora G, eds. Advances in Software Engineering and Knowledge Engineering. New Jersey: World Scientific Publishing Co., 1993.

[5] Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M, Wrote; Ben KR, Guo FL, Zhao A, *et al*., Trans. Pattern-Oriented Software Architecture, Vol.1: A System of Patterms. Beijing: China Machine Press, 2003 (in Chinese).

[6] Schmidt D, Stal M, Rohnert H, Buschmann F, Wrote; Zhang ZX, Ren XW, Xiao B, *et al*., Trans. Pattern-Oriented Software Architecture, Vol.2: Pattern for Concurrent and Networked Objects. Beijing: China Machine Press, 2003 (in Chinese).

[7] Kircher M, Jain P, Wrote; Bao ZY, Trans. Pattern-Oriented Software Architecture, Vol.3: Patterns for Resource Management. Beijing: China Machine Press, 2005 (in Chinese).

[8] Gamma E, Helm R, Vissides J, Johnson R Wrote; Li YJ, Ma XX, Cai M, Liu JZ, Trans. Design Patterns: Elements of Reusable Object-Oriented Software. Beijing: China Machine Press, 2000. 16−17 (in Chinese).

[9] Frankel DS Wrote; Bao ZY, Trans. Model Driven Architecture: Applying MDA to Enterprise Computing. Beijing: Posts & Telecom Press, 2003. 5−8 (in Chinese).

[10] Licberherr KJ, *et al*. Demeter: Aspect-Oriented software development. 2004. http://www.ccs.neu.edu/research/demeter/

[11] Highsmith J, Write; Qian L, Trans. Adaptive Software Development. Beijing: Tsinghua University Press, 2003 (in Chinese).

[12] Huang SX, Fan YS, Zhao Y. Research on generic adaptive software architecture style. Journal of Software, 2006,17(6):1338−1348 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1338.htm [doi: 10.1360/jos171338]

[13] Cheng SW, Garlan D, Schmerl B, Sousa JP, Spitznagel B, Steenkiste P, Hu N. Software Architecture-Based Adaptation for Pervasive Systems. Berlin: Springer-Verlag, 2002.

[14] Jacobson I, Booch G, Rumbaugh J. The Unified Software Development Process. Addison-Wesley, 1999.

[15] Clements P, Kazman R, Klein M, Wrote; Sun XT, Zhu WD, Zhao K, Trans. Evaluating Software Architectures. Beijing: Tsinghua University Press, 2002 (in Chinese).

[16] Ziv H, Richardson DJ. Constructing Bayesian-network models of software testing and maintenance uncertainties. In: Li YJ, Ma XX, Cai M, eds. Proc. of the Int'l Conf. on Software Maintenance. 1997.

[17] Kan SH, Wrote; Wu MH, Ying J, *et al*., Trans. Metrics and Models in Software Quality Engineering. 2nd ed., Beijing: Electronic Industry Press, 2004 (in Chinese).

[18] Gao H, Zhang L. Research on flexibility metrics in software architecture level. Computer Science, 2008,35(4):259−264 (in Chinese).

[19] Gao H, Zhang L. Research on software architecture level structure metrics. Computer Engineering and Applications, 2007,43(24): 19−23 (in Chinese).

[20] Cooper G, Herskovits E. A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 1992,9: 309−347.

[21] Microsoft Research. MSBNx. 2006. http://research.microsoft.com/adapt/MSBNx/

[22] Kadie CM, Hovel D, Horvitz E. MSBNx: A component-centric toolkit for modeling and inference with bayesian networks. Microsoft Research, 2001.

[23] KSU Probabilistic Reasoning Group. BNJ. 2004. http://bnj.sourceforge.net/

[24] Gamma E, Beck K, Wrote; Xiong J, Trans. Contributing to Eclipse Principles, Patterns, and Plug-Ins. Beijing: China Electric Power Press, 2004 (in Chinese).

[25] Gao H, Zhang L, Fan ZQ. Software architecture description technique based on template. Journal of Beijing University of Aeronautics and Astronautics, 2008,34(1):122−126 (in Chinese).

[26] Fenton N, Neil M. New directions in software metrics. 1999. http://www.dcs.qmul.ac.uk/~norman/papers/new_directions_metrics/start.htm [doi: 10.1109/32.815326]

[27] Fenton N, Neil M. A critique of software defect prediction models. IEEE Trans. on Software Engineering, 1999,25(5):675−689. [doi: 10.1109/TSE.2002.1010058]

[28] Yacoub SM, Ammar HH. A methodology for architecture-level reliability risk analysis. IEEE Trans. on Software Engineering, 2002,28(6):529−547.

[29] Gurp JV, Bosch J. Automating software architecture assessment. In: Proc. of the 9th Nordic Workshop on Programming and Software Development Environment Research. 2000.

[30] Gurp JV, Bosch J. SAABNet: Managing qualitative knowledge in software architecture assessment. In: Proc. of the 2000 IEEE Conf. on Engineering of Computer Based Systems. 2000.

附中文参考文献:

[5] Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M,著;贲可荣,郭富亮,赵皑,等,译.面向模式的软件体系结构,卷 1:模式系统.北京:机械工业出版社,2003.

[6] Schmidt D, Stal M, Rohnert H, Buschmann F,著;张志祥,任雄伟,肖斌,等,译.面向模式的软件体系结构,卷 2:用于并发和网络化对象的模式.北京:机械工业出版社,2003.

[7] Kircher M, Jain P,著;鲍志云,译.面向模式的软件体系结构,卷 3:用于资源管理的模式.北京:机械工业出版社,2005.

[8] Gamma E, Helm R, Johnson R, Vissides J, Johnson R,著;李英军,马晓星,蔡敏,刘建中,译.设计模式——可复用面向对象软件的基础.北京:机械工业出版社,2000.16−17.

[9] Frankel DS,著;鲍志云,译.应用 MDA.北京:人民邮电出版社,2003.5−8.

[11] Highsmith J,著;钱岭,译.自适应软件开发.北京:清华大学出版社,2003.

[12] 黄双喜,范玉顺,赵彧.一类通用的适应性软件体系结构风格研究.软件学报,2006,17(6):1338−1348. http://www.jos.org.cn/1000-9825/17/1338.htm [doi: 10.1360/jos171338]

[15] Clements P, Kazman R, Klein M,著;孙学涛,朱卫东,赵凯,译.软件构架评估.北京:清华大学出版社,2002.

[17] Kan SH,著;吴明晖,应晶,等,译.软件质量工程——度量与模型.第 2 版,北京:电子工业出版社,2004.

[18] 高晖,张莉.软件体系结构层次的适应性度量技术研究.计算机科学,2008,35(4):259−264.

[19] 高晖,张莉.软件体系结构层次的结构度量技术研究.计算机工程与应用,2007,43(24):19−23.

[24] Gamma E, Beck K,著;熊节,译. Contributing to Eclipse 中文版.北京:中国电力出版社,2004.

[25] 高晖,张莉,樊志强.基于模板的软件体系结构描述技术.北京航空航天大学学报,2008,34(1):122−126.

高晖(1977－),男,四川成都人,博士,工程师,主要研究领域为软件体系结构,MDA 相关技术.

张莉(1968－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,需求工程,软件体系结构,过程建模和优化.

李琳(1984－),女,博士生,主要研究领域为软件体系结构,软件度量.