文章编号:1000-5862(2021)01-0055-05

基于多维属性演化树的软件演化风格匹配方法研究

钟林辉,齐 杰,叶海涛,莫俊杰
(江西师范大学计算机信息工程学院,江西 南昌 330022)

摘要:为了提高软件企业的过程能力,该文提出一种基于多维属性演化树的软件演化匹配方法,利用软件的每个版本包含的原子构件个数、原子构件大小、软件体系结构大小、有效代码行数、java 文件数等属性,再通过树匹配的方法找出特定类型项目的软件演化风格,优化软件企业演化过程,为提高软件企业的过程能力提供条件.

关键词:软件体系结构;演化树;软件演化;遗传算法
中图分类号:TP 311 文献标志码:A DOI:10.16357/j.cnki.issn1000-5862.2021.01.08

0 引言

随着社会信息化程度越来越高,软件的功能越来越强大,但随着软件功能的不断增加,软件也变得更加复杂,参与开发的人员也在不断增加,导致软件产品对软件开发人员的依赖性逐渐增大,使得软件项目过程改进难度增大,如何实现过程改进的最佳效益成为急待解决的问题[1-8]. 为了解决这个急迫的问题,研究人员已经花费了大量精力和时间进行研究[9-12],如 CMMI(软件能力成熟度集成模型)是一种用于评价软件企业的开发过程能力并且帮助其改善软件质量的方法. CMMI 分 5 个等级,即完成级、管理级、定义级、量化管理级和优化级,每个等级包含的过程域共同形成了软件过程能力. 企业能够充分利用信息资料,对企业在项目实施的过程中可能出现的次品予以预防,能够主动地改善流程,改进软件过程能力,使现有的过程更加高效,解决软件项目过程难度增加的问题.

软件演化信息是软件变化过程中存储或蕴含的有用信息,在对软件演化的研究过程中,笔者发现同一类型软件在演化过程中存在特定的演化风格,找出并使用同类软件的演化风格有利于开发人员借鉴同类型软件的开发经验,充分利用这些有用的信息优化同类软件的演化过程,提高软件企业的过程能力.

为了找出同类软件的演化风格,本文以软件演化树和软件系统的属性信息为研究对象,提出了基于多维属性演化树的软件演化风格匹配方法.

1 相关研究

通过软件复用能够有效地提高软件企业的过程能力,软件复用一般包括软件产品的复用和过程复用 2 种. 对于软件产品的复用,有许多学者进行了深入研究,如文献[4]提出了一种支持领域特性的 Web 服务组装方法,在进行系统开发时,充分考虑领域特性可以有效地利用领域工程所得到的制品,提高系统的开发效率. 文献[5]围绕复用构件的通用性、可复用能力,建立可复用的应用程序生成器,提高了开发效率和软件质量. 文献[12]提出一种基于服务组装的可复用、可定制的 SaaS 软件开发方法,针对不同租户的共性需求提供一个抽象服务组装模型,根据租户的个性化需求派生不同的流程实例. 优化开发过程并降低软件成本.

对于过程复用,其内容涉及过程建模、过程实施、过程度量、过程评价和过程优化等[13-18]. 目前,过程复用主要是基于模型的方法,如文献[11]提出一种基于复用的软件过程改进方法,该方法以可复用的软件过程为基础,以过程资产库为中心,基于项目过程的不断优化调整构成,使得软件开发过程能根据软件开发机构自身特点按照一定标准进行,有

收稿日期:2020-08-12
基金项目:国家自然科学基金(61462040,61262015,61662032,61762049),江西师范大学教改课题(JXSDJG2044)和江西省自然科学基金(20142BAB207027,20142BAB207027,20142BAB207027)资助项目.
作者简介:钟林辉(1974-),男,江西赣州人,教授,博士,主要从事软件体系结构、构件化软件、软件维护与软件演化研究. E-mail:shiningto@jxnu.edu.cn

效地提高软件过程的可控性和项目的成功率. 文献[5,6]对特征模型的元模型进行了扩展,极大地丰富了特征模型中的元素种类,提升了特征模型的描述能力. 文献[5,8]则基于上述基本思想实现了手机软件产品的自动导出,采用软件过程建模的方法优化软件过程提高软件生产效率. 文献[7]提出安卓应用界面和业务逻辑的结构模型,以统一的方式描述安卓应用的界面元素、业务逻辑以及2者的关联关系,通过建立软件过程模型设计相应的工具,提高了安卓应用开发效率. 文献[9]提出了一种基于控件和XML配置技术的可定制的软件开发方案,通过运行时的XML数据提取、分析、控件及其控制代码的扩展,实现软件的定制,降低软件的开发难度,适应不断变化的用户需求. 但在上述过程复用的研究中,对优化软件演化过程的研究较少,对软件演化历史及其在演化历史中软件固有属性变化趋势的研究明显不足.

因此,本文提出了一种基于多维属性演化树的软件过程演化风格的查询方法,通过找出同类型软件的特有演化风格帮助企业优化软件演化过程,提高软件企业的过程能力度.

2 演化风格定义及相似性度量

2.1 软件演化风格定义

将软件演化风格定义为特定领域内的一组软件演化过程的共性. 软件演化风格表现为领域内的一组软件演化过程在其结构上的相似性.

定义1 软件演化过程 T 体现为软件的变化历史,可以表示为一棵演化树. 其中树中不同节点表示软件的不同版本,节点的标识为软件的版本号,树节点的分支则表示一次变化.

定义2 软件演化风格T-Style亦可以表示为一棵树,对于一组软件的演化过程 T_1、T_2、T_3,软件演化风格T-Style是 T_1、T_2、T_3 中结构相似的子树,表示一组软件有着相似的演化过程,代表了一组软件演化过程的共性.

2.2 带属性集合的软件演化树

软件系统的演化历史可以用演化树表示,本文获取了github上开源软件的多个版本,通过逆向的方法得到软件系统的各个版本的属性,其属性包括原子构件个数、原子构件大小、软件体系结构变化度量、有效代码行数等属性信息. 每个版本作为节点并按照版本号的先后顺序组成软件演化树,节点的标识为github中的版本号,每个节点带有逆向得到的多维属性的软件演化树.

2.3 相似度度量

在软件演化树进行节点对应时,需对不同演化树的节点作一个相似度度量. 这里的相似度度量比较的是2个版本之间属性数组之间的相似度. 在本文中,软件版本属性数组包括原子构件个数、原子构件大小、软件体系结构、有效代码行数、java文件数等5个属性. 利用文献[18]中选择余弦公式来度量2个版本间的相似度,计算公式为

$$S_{im}(S_1,S_2)=\frac{S_1S_2}{\|S_1\|\ \|S_2\|}=\frac{\sum_{i=1}^{n}a_ib_i}{\sqrt{\sum_{i=1}^{n}a_i^2\sum_{i=1}^{n}b_i^2}},$$

其中 S_1、S_2 分别表示版本1和版本2的属性向量,2个构件版本间的相似度的取值区间为(0,1]. 若相似度的值越接近于1则这2个版本越相似,若相似度越接近于0则这2个版本相似程度越低.

3 软件演化风格匹配方法

软件演化风格匹配方法是通过树匹配的方式找出不同软件之间具有相同结构的演化树子树,即特定的软件演化风格. 首先要确定不同演化树的节点对应关系并选择树匹配模型,然后验证节点在对应后的演化树与匹配对象的演化树结构是否一致.

3.1 树匹配模型建立

文献[1]提出了5种树匹配模型,即子树匹配(Ms)、区域匹配(Mr)、包容匹配(Me)、强约束包容匹配(Mse)和弱约束包容匹配(Mle). 每种匹配模式都有不同的匹配精度,其匹配精度由大到小的排列顺序为子树匹配(Ms)、区域匹配(Mr)、强约束的包容匹配(Mse)、弱约束的包容匹配(Mle)和包容匹配(Me). 在树匹配模型的节点上带有原子构件个数、原子构件大小、软件体系结构大小、有效代码行数、java文件数共5维度的属性.

3.2 演化树节点关系对应

基于多维属性演化树的软件演化风格匹配是用树匹配的方式找出2个演化树结构相同的一组演化树. 2个不同软件取连续版本得到软件演化树,并让演化树中所有节点关联上相应版本的5个维度上的属性. 将一个软件演化树中的所有节点与另外一个软件演化树中的所有节点进行相似性计算,通过相似性的大小可以得到一个软件演化树中的节点与另一个软件演化树中节点相似度最高的节点,这2个不同演化树的节点组成节点对. 判断节点对中的节点是否为有效等价节点需要设置一个相似度阈值作

为标准,若相似度超过设定的阈值则视为节点对中的 2 个节点是有效等价节点. 阈值的设定会影响匹配的精度,阈值设定越高,匹配得到的结果越准确,但匹配成功的概率就越小,匹配的成本就越高.

因此,若设定节点对的相似度大于 80%,则视为有效的等价节点. 如图 1 所示,以 2 棵演化树示例,设 TreeQ 为需要匹配的演化树,TreeT 为测试数据集中的演化树,首先需要将演化树 TreeQ 中的节点与 TreeT 中的每一个节点进行相似性计算,计算结果如表 1 所示.

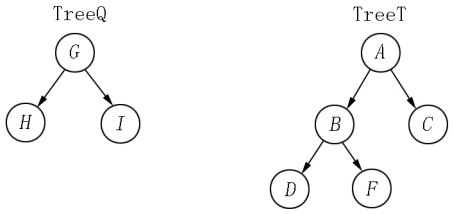

图 1　演化树示例

表 1　节点相似性计算结果

类型	A	B	C	D	F
G	0.982 1	0.942 3	0.912 3	0.723 12	0.813 4
H	0.324 1	0.931 2	0.823 4	0.882 34	0.963 1
I	0.812 3	0.873 1	0.903 1	0.931 24	0.643 1

分析表 1 数据可知,G 节点的有效等价节点有 4 个,分别是 A、B、C 和 F 节点,D 节点与 G 节点相似度小于设定的阈值,所以为无效等价节点. 在有效等价的 4 个节点中 G 节点与 A 节点相似度最高,所以 G 节点与 A 节点为最佳等价节点. 同理,H 节点有 4 个有效等价节点,最佳等效节点为 F 节点. I 节点有 4 个有效等价节点,最佳等效节点为 D. 节点对应之后的 TreeQ 可等价地看成 TreeQ_B 所表示的演化树(见图 2).

图 2　节点对应之后的演化树

3.3　软件演化树匹配算法

根据上述树匹配建模和建立的树节点的对应关系,对 2 棵演化树进行匹配. 使用遗传算法和树匹配算法相结合,改善由于节点增加导致匹配速度变慢的情况. 伪代码如算法 1 所示(输入为待匹配的演化过程 Q、历史演化过程 T,输出为符合匹配模型的演化过程 $G_{Q\to T}$).

算法 1　软件演化树匹配算法.

输入: $Q=(V,E,\text{root}(Q))$, $T=(W,F,\text{root}(T))$.

输出: $G_{Q\to T}(\text{Emi})$.

1) Count = 0;
2) fitness = + ∞;
3) intial population from Q;
4) Do while
6) Select a subset X_{sub} of population
7) Do while
9) Select a subset T_{sub} of from $T\prod_{q_i\in X_{sub}} M(q_i)$;
10) If Validate($Q_{sub}\to T_{sub}, M_i$) Then
12) min cost =
13) If(fitness > cost) {fitness = min cost}
14) Else{population. remove(X_{sub})
15) End If
16) End While(Try out all possible T_{sub})
17) Crossover( )
18) Mutation( )
19) Update population
20) if(fitness does not change)
21) Count + +
22) End If
23) End While(Count < 50).

算法 1 使用遗传算法优化了子树枚举和匹配的过程,定义树匹配得到的编辑代价作为适应度,通过减少匹配的次数降低由于树节点增加对匹配时间的影响,使得匹配算法能在较短的时间内完成较大规模节点的树匹配.

首先,从树 Q 中得到一个包含子树的子树集作为遗传算法的初始种群,并设置适应度的初始值,然后用初始种群中的子树与树 T 枚举的子树进行匹配,若符合区域匹配模型的子树则进一步计算编辑代价,将计算的编辑代价与设置的适应度进行比较,若适应度大于编辑代价,则将此子树从子树集中删除;若适应度小于编辑代价,则将适应度更新为当前的匹配代价,匹配的子树保留在子树集中,之后循环进行交叉变异选择操作,当适应度在 50 次循环中均不再变化时将满足停止条件.

4　工具支持

为了支持多维属性演化树的软件演化风格匹配,本文开发了相应的支持工具. 系统主要分为 6 个模块,他们分别是前端演化过程信息输入模块、演化树节点相似度度量模块、演化树编码模块、子树枚举模块、演化树匹配模块、前端展示模块. 系统结构如图 3 所示.

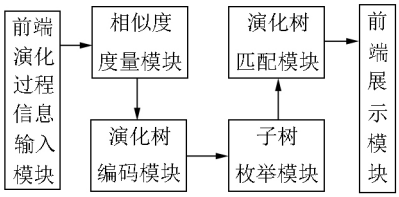

图 3 软件演化树匹配系统结构

(i) 前端演化过程信息输入模块. 该模块为用户提供待匹配演化过程信息的输入功能,首先将待匹配演化过程按照版本的先后顺序以版本号的形式组成字符串,然后将待匹配软件的演化过程信息的字符串还原为演化树,并将逆向得到的 5 个维度的属性关联到相应节点上,得到多维属性的软件演化树并提供给其他模块使用.

(ii) 演化树节点相似度度量模块. 通过比较当前软件演化树和已有的演化树中节点的相似性,找出当前软件的演化树中节点与已有演化树节点之间的实际对应关系.

(iii) 演化树编码模块. 使用二进制的方式将还原的演化树进行编码,为后续遗传算法匹配做准备.

(iv) 子树枚举模块. 该模块的主要功能是将演化树的子树进行枚举,产生待分析软件演化树的初始子树集作为在使用遗传算法时的初始种群,由于待分析软件演化树在经过演化树编码模块处理后,其表示方式为二进制,因此子树集中的子树表示方式与待分析软件演化树编码后的表现方式均为二进制.

(v) 演化树匹配模块. 该模块是基于遗传算法的演化树匹配的核心模块,包括树匹配模块和相似度比较模块 2 个子模块. 利用枚举得到的子树集作为初始种群,将编辑代价定义为适应度,利用遗传算法对种群进行个体评价、选择运算、交叉运算、变异运算操作,并选择出编辑代价最小的一组子树集作为最优解.

(vi) 前端展示模块. 该模块将符合匹配规则的子树按照在查询时输入的字符串形式,将子树转化为字符串信息输出到前台界面进行展示.

5 实验

选取 6 个开源系统的源代码作为实验数据,其中 Cassandra 选取 78 个版本、Hbase 选取 36 个版本、Hadoop 选取 52 个版本、zookeeper 选取 23 个版本、hive 选取 30 个版本、openjap 选取 6 个版本. 将 Cassandra、Hbase、Hadoop、zookeeper、hive 共 5 个系统的演化树信息作为测试数据集,把 openjap 的演化树作为测试对象,系统版本号是 github 中的版本号. 通过与测试数据集的比较得到与 openjap 演化树分别在 5 种匹配模式下相匹配的演化子树并对其结果进行分析.

首先通过 6 个系统的源代码进行逆向分析获得系统每个版本的原子构件个数、原子构件大小总和、软件体系结构大小、有效代码行数、java 文件数. 然后,在 github 中得到该软件的软件演化树,其节点的编号为 github 中设定的版本号,并将节点关联上逆向得到的 5 维属性.

在得到 6 棵多维属性软件演化树后,使用带属性集合的软件演化树匹配原型系统进行树匹配分析,使用原型工具分别进行子树匹配(Ms)、区域匹配(Mr)、强约束的包容匹配(Mse)、弱约束的包容匹配(Mle)和包容匹配(Me) 5 种模式的匹配,演化树匹配的结果如表 2 所示.

表 2 演化树匹配结果

类型	Ms	Mr	Mse	Mle	Me
Cassandra	13	27	45	47	72
Hbase	8	19	29	29	56
Hadoop	11	22	31	32	60
zookeeper	4	12	19	21	37
hive	4	15	22	25	31

由表 2 可知,包容匹配(Me)所得匹配结果个数最高,最低的匹配模式为子树匹配(Ms). 但通过对 5 种匹配得到子树进行分析知,子树匹配(Ms)得到的演化子树的版本个数的匹配精度是最高的,包容匹配(Me)得到的结果精度是最低的. 样本演化树与匹配所得演化树节点个数比的结果如图 4 所示,在 5 种匹配模式的软件演化树匹配中,子树匹配(Ms)→区域匹配(Mr)→强约束的包容匹配(Mse)→弱约束的包容匹配(Mle)→包容匹配(Me)在版本个数的匹配精度中是逐渐增高的一个过程. 匹配成功的概率则逐步降低.

通过对 5 种匹配得到子树所代表软件版本进行逆向,得到其演化树所有节点的 5 维属性结果并进行分析,其中子树匹配模式下匹配得到的软件演化子树其代表的版本个数、属性变化趋势与需要匹配的演化树完全一致.

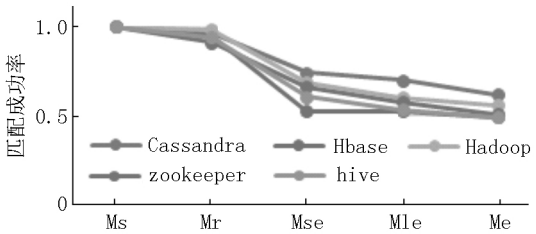

图 4 样本演化树与匹配所得演化树节点个数比

6 结束语

软件演化树对软件演化的历史进行刻画,通过

软件演化树找出其特定类型软件的演化风格有利于软件开发经验复用，利用这些有用的信息优化同类软件的演化过程，提高软件企业的过程能力. 为了达到上述目的，提出了基于多维属性演化树的软件演化匹配方法，以 6 个开源系统作为实验对象，从软件每个版本的原子构件的个数、原子构件大小总和、软件体系结构大小、有效代码行数、java 文件数 5 个属性的角度进行分析. 实验效果表明：通过上述方法可以在不同精度要求下找出相匹配的软件演化树子树，为寻找特定类型软件的演化风格提供有效的方法，为提高软件企业的过程能力提供有利条件.

7　参考文献

[1] 王渊峰，薛云皎，张涌，等. 刻面分类构件的匹配模型[J]. 软件学报，2002，14(3)：402-408.

[2] Griss M L，Favaro J，d´Alessandro M. Integrating feature modeling with the RSEB [EB/OL]. [2020-07-16]. https://www.researchgate.net/profile/John_Favaro/publication/2914328_Integrating_Feature_Modeling_with_the_RSEB/links/0c9605174f05b8e9db000000/Integrating-Feature-Modeling-with-the-RSEB.pdf.

[3] Botterweck G，Lee K，Thiel S. Automating product derivation in software product line engineering [EB/OL]. [2020-07-16]. http://cs.emis.de/LNI/Proceedings/Proceedings143/article4639.html.

[4] 赵俊峰，谢冰，张路，等. 一种支持领域特性的 Web 服务组装方法[J]. 计算机学报，2005，28(4)：732-738.

[5] 吴昌钱. 基于元数据的软件复用构件[J]. 西安文理学院学报，2015，18(4)：57-60.

[6] Zhang W. Research on feature-oriented domain modeling [D]. Peking: Peking University，2006.

[7] 朱亚迪，吴毅坚，赵文耘. 基于代码片段复用的安卓应用组装技术研究[J]. 计算机应用与软件，2016，33(11)：164-168.

[8] Chaves F J E. Dynamic and automated product derivation for consumer electronics software applications[J]. IEEE Transactions on Consumer Electronics，2013，59(4)：883-891.

[9] 李广强，吴伟民，赖天武，等. 基于控件和 XML 的可定制软件开发方案[J]. 计算机工程，2007，33(2)：267-282.

[10] 王红. 面向模式的支持多粒度服务复用的软件开发[J]. 计算机应用，2011，31(1)：132-137.

[11] 徐如志，都艺兵，于华，等. 基于复用的软件过程改进方法[J]. 计算机科学，2006，33(6)：251-254.

[12] 孙昌爱，张在兴，张鑫. 一种基于可变性模型的可复用与可定制 SaaS 软件开发方法[J]. 软件学报，2018，29(11)：3435-3454.

[13] 韩恺，岳丽华，龚育昌. 基于上下文的异构文档类型定义匹配[J]. 小型微型计算机系统，2005，26(2)：256-260.

[14] 钟林辉，侯长源，宗洪雁，等. 构件化软件演化信息及演化相似性度量技术研究[J]. 计算机应用研究，2015，32(5)：1399-1416.

[15] 钟林辉，宗洪雁. 基于本体的构件化软件演化信息获取及度量研究[J]. 计算机科学，2015，42(1)：196-231.

[16] 王渊峰，张涌，任洪敏，等. 基于刻面描述的构件检索[J]. 软件学报，2012，13(8)：1546-1551.

[17] Francisco S，James A J. Fuzzy fine-grained code-history analysis [EB/OL]. [2020-07-16]. https://dl.acm.org/doi/pdf/10.1109/ICSE.2017.74? download=true.

[18] 钟林辉，李俊杰，夏鲸，等. 基于多维属性的构件化软件演化相似性度量方法研究[J]. 计算机科学，2016，43(11)：449-505.

The Study on the Method for Matching the Software Evolutionary Style Based on Multi-Dimensional Evolutionary Tree

ZHONG Linhui，QI Jie，YE Haitao，MO Junjie

(College of Computer and Information Engineering，Jiangxi Normal University，Nanchang Jiangxi 330022，China)

Abstract: To improve the process of software projects for enterprises，a software evolution matching method is put forward based on software evolution tree，by multi-dimensional attributes as the amounts and the sizes of atomic components，the sizes of software architectures，effective lines of codes and the amounts of Java files of every software version. Through the method，the software evolution style of a specific project type can be found and the software evolutionary process for enterprises can be optimized as well，which provide favorable conditions for the process capability improvement of software enterprises.

Key words: software architecture; evolutionary tree; software evolution; genetic algorithm

(责任编辑：冉小晓)