复杂软件系统健康状态智能感知与诊断模型

王 森[1],王 煜[2],宁德军[2]
(1. 同济大学电子与信息工程学院,上海 201804;2. 中国科学院上海高等研究院,上海 201210)

摘 要: 随着工业物联网和人工智能技术的迅猛发展,各种复杂软件系统(Complex Software System, CSS)日趋盛行,成为最重要的软件系统开发范式之一,其固有的成长性构造和适应性演化性质要求CSS必须能够实时感知和诊断自身的健康状态,确保其适应性演化过程中的质量.本文采用特征工程和存储库数据挖掘技术,对影响开源CSS健康状态的特征进行分析,建立了一个数据驱动的实时、客观地反映开源CSS健康状态的自感知模型,并进一步借鉴质量控制图的思想,定义了能够辅助开源CSS故障诊断的自诊断模型.最后,通过对比实验,证明了本文提出的模型因为全面综合了软件开发过程的绝大多数特征,能够更加全面和有效地评价软件的健康状态.

关键词: 复杂软件系统; 存储库数据挖掘; 自感知模型; 自诊断模型
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112(2021)09-1799-10
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20190396

Intelligent Perception and Diagnosis Model for Health Status of Complex Software System

WANG Sen[1], WANG Yu[2], NING De-jun[2]
(1. *School of Electronics and Information Engineering, Tongji University, Shanghai* 201804, *China*;
2. *Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai* 201210, *China*)

Abstract: With the rapid development of Industry Internet of Things and AI technology, various complex software systems (CSS) are becoming more and more popular, and becoming one of the most important software development paradigms. Its inherent growth construction and adaptive evolution require CSS to be able to perceive and diagnose its own health status in real time, so as to ensure the quality of its adaptive evolution. The paper uses feature engineering and mining software repositories (MSR) technology to analyze the features that affect the health of open source CSS, and establishes a data driven perception model that can reflect the health status of open source CSS in real time and objectively. Furthermore, a self-diagnosis model that can assist open source CSS fault diagnosis is defined with reference to the quality control chart. Finally, through the model comparison experiment, it is proved that our model can evaluate the health of software more comprehensively and effectively because it integrates most feature of software development process.

Key words: complex software system; mining software repositories; self-perception model; self-diagnosis model

1 引言

当前,人机物深度融合的智能化时代,催生了各种智能化的复杂软件系统.复杂软件系统(Complex Software System, CSS)[1]是指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,它既包括智慧城市领域的智能服务系统,也包括工业互联网领域的信息物理系统(Cyber Physical System, CPS)[2].区别于传统的软件,它们多以智能化服务和社会分工体系[3]创新为目标,以数据为主要生产要素,呈现出成员异构性、行为涌现性、边界开放性和持续变化性等典型特征,具有成长性构造和适应性演化的基本性质[1],因此,要求CSS的开发团队必须能够在适应性演进过程中快速感知CSS的健康状态,诊断自身可能存在的健康问题并持续改进.因此,CSS健康状态的快速感知和诊断能力,正成为成功开发CSS的基本需求.

随着CSS的快速发展,学术界开始关注其成长性构

收稿日期:2019-04-15;修回日期:2020-12-31;责任编辑:梅志强
基金项目:上海市经济和信息委员会工业互联网创新发展专项资金项目(No. 2020-GYHLW-02010);上海化学工业区公共事务中心上海化学工业区智慧决策平台项目(No. E0420S1);竞技体育高水平运动队人工智能辅助训练系统项目(No. DQ200966-00)

造和适应性演化过程的自感知能力的研究. 自适应过程是指能够根据对环境和系统本身的感知,来改变自身行为或结构,以更好地实现其目标[4]. 软件开发过程本身就是一个不断适应软件开发需求的过程[5],而实现软件的自适应方法主要包括基于控制论的MAPE-K自治计算模型[6,7]和基于决策论的数据驱动自适应模型[8]. 为了深入了解软件开发过程中的各种特征对项目健康状态的影响,一些早期学者使用特征建模的方法[9,10]分析影响软件项目健康的关键因素,近年来随着开源软件开发范式的日益繁荣,为数据驱动的软件健康度的感知诊断提供了大量存储库数据,很多学者通过存储库数据挖掘(Mining Software Repositories, MSR)的方法[11,12]分析影响项目健康状况的各种特征要素.

本文将以开源社区存储库数据挖掘为手段,以自适应系统理论为指导,研究CSS软件健康状况自感知和自诊断模型,帮助CSS软件开发团队自动地感知和智能地诊断自身系统健康状态.

2 CSS开发和演进过程分析

2.1 理解CSS开发和演进过程

Github社区软件开发活动既包括由创建分支、新增提交、提交合并请求、讨论和复审、合并和部署等五个基本开发活动组成的基本开发工作流,也包括诸如关注、加星、注释等开源软件社区生态系统活动[13]. 与此相似,CSS的开发和演化过程可以抽象成以软件系统为核心,由用户需求、开发团队、开发过程、开发平台以及用户共同组成的有机体,如图1所示,输入的是用户需求和团队知识,输出的是供用户使用的软件系统,有机体的核心工作是由开发者和用户基于社区协作平台开展的协作开发活动,它会持续推动CSS的成长性构造和适应性演化过程.

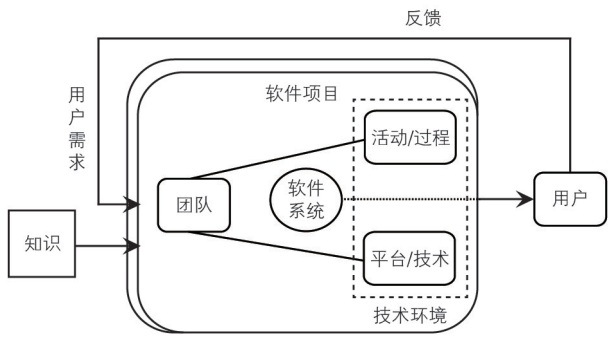

图1 CSS开发生态系统概念模型

因此,从系统论的角度看,CSS开发社区由多个同类的软件项目组成,每个软件项目可以被定义为由软件系统、团队、过程和平台等要素以一定结构形式联结构成的具有某种功能的有机整体,可以形式化地表示如下.

定义1 CSS项目:$\text{Project}_{\text{CSS}}=\langle S,F\rangle=\langle E,A,D\rangle$;其中$S$为结构,且$S=\langle E,D\rangle$,表示若干要素$E$以一定结构形式联结,相关的要素之间及其与外部环境的交互,会以数据D的方式得以记录;F为功能,且$F=<A,D>$,表示系统中的联结的元素通过协作活动A实现某种功能F,整个协作过程会以数据D的方式得以记录.

定义2 CSS项目要素:$E=\{e_{\text{team}},e_{\text{software}},e_{\text{process}},e_{\text{platform}},e_{\text{user}}\}$;其中,$e_{\text{team}}$包括CSS的项目经理、核心团队和扩展合作团队[5];e_{software}就是指CSS,包括代码、文档及相关知识等;e_{process}特指CSS协作开发所用的软件过程;e_{platform}代表了CSS软件协同开发平台;e_{user}主要指参与了社区互动的CSS用户.

定义3 CSS项目活动:$A=\{a_{\text{dev}},a_{\text{sn}}\}$;其中,$a_{\text{dev}}$为社区内置的基本软件开发活动,活动序列可以构成一个具体的开发过程(process),在Github中的基本软件开发过程被称作是Github Flow,如图2所示;a_{sn}表示社会化协作活动,在Github中包括评论(comments)、关注(watch)、加星(star)、复制软件(fork)和论坛讨论等活动.

图2 Github开发过程

定义4 CSS项目数据:$D=\{d_{\text{team}},d_{\text{software}},d_{\text{process}},d_{\text{platform}},d_{\text{user}}\}$;其中,$d_{\text{team}}$为CSS社区团队活动的数据,$d_{\text{software}}$是围绕CSS产生的相关数据,$d_{\text{process}}$主要包括基本协作开发过程留下的数据,$d_{\text{platform}}$主要包括基于平台的社区协作活动留下的数据,$d_{\text{user}}$是用户参与过程中留下的数据.

因此,理论上可以认为Github社区就是由很多的CSS项目以及非CSS软件构成的生态环境,而非CSS软件包括了那些成长性构造所依赖的潜在优秀软件.

由定义1可知,若要实现对开源软件健康状态的全面感知,则必须能够适时感知主要项目元素和项目活动的状态,并进一步推理出开源软件的健康状况,由此,给出如下的开源软件健康状态感知模型定义.

2.2 开源软件存储库数据挖掘分析框架

考虑到开源软件开发和演进过程中所有项目元素(E)的多数项目活动(A)都会以项目数据(D)的方式记录在存储库中,如图3所示,本文通过存储库数据挖掘和分析建模方法,建立CSS软件健康状态的自感知模型和故障自诊断模型,整个过程包括数据准备、感知模型定义、诊断模型定义3个主要步骤.

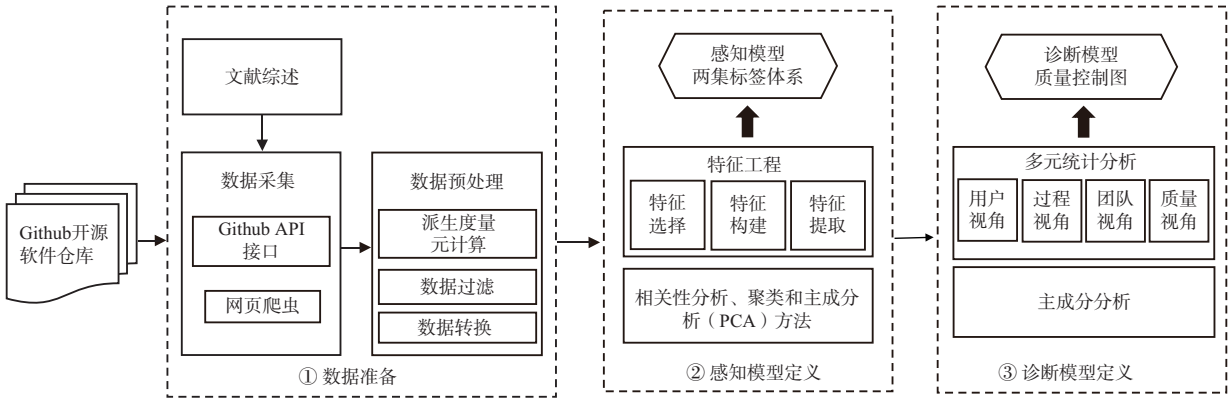

图3　CSS自感知自诊断模型研究框架

3　CSS自感知模型定义

3.1　数据准备

本文利用Github API[14]和网页爬虫进行相关数据采集,采集数据中共包含约27万个项目的54个属性数据. 其中,通过抓取follower数排名前1000人的项目,共得到约32000条记录. 随后通过在这1000名开发者的关注者群体中随机采样,抽取了约23万多个项目. 本次数据集中星数超过1000,星数超过500,星数超过100的开源项目占比分别为85. 32%,81. 13%,59. 72%,充分体现了Github社区中大部分优质项目的情况,具有代表性.

数据预处理过程主要包括数据筛选、数据转换和派生度量元计算.

(1)　数据筛选

首先通过异常值处理,如过滤空值,过滤异常数据等处理手段,对原始数据进行数据清洗,保留下90956条存储库记录;其次,为了过滤出真正的软件开发项目,从数据集中过滤掉了代码行数小于500、或者合并请求关闭数等于零、或者问题关闭数等于零的项目,共得到47530条开源项目的信息;最后,过滤掉缺陷关闭数、里程碑数和提交评论数为零和缺陷关闭时间异常的项目,最终得到17478条开源项目数据作为实验数据集.

(2)　数据转换

对实验数据集进行数据审核时,发现绝大多数的属性都呈正偏分布,且偏度和峰度都较大,很多记录的众数为0,甚至少量记录的中位数也为0. 故对上述属性进行Log10变换,使其尽可能接近正态分布,以便进一步统计分析.

(3)　派生度量元计算

借鉴软件工程知识和文献[11,12]分析结果,部分基本度量元被转换成为速度型派生度量元、人均型派生度量元和缺陷派生度量元. 例如,为了尽可能消除项目生存周期的影响,更好的从团队效率角度感知开源软件,生成了速度型派生度量元,如外部发布心跳表示平均每月的发布版本数;为了消除团队规模的影响,更好地从贡献者个人角度去感知开源软件,生成了人均型派生度量元,如人均代码行;为了能够得到感知开源软件质量的缺陷信息,基于Github中拥有“defect”、“bug”等问题标签的项目,生成了缺陷关闭率,缺陷关闭速度等派生度量元.

3.2　特征工程

为了更加科学地了解和研究各种特征属性之间的相关性及其对开源软件健康状况影响,本文采用了特征工程方法选择、构建和提取能更好地支持分析目标的特征集.

3.2.1　特征选择

特征选择就是从特征集合中挑选一组最具统计意义的特征子集. 在特征选择过程中,首先明确它要解决的基本问题:

Q1:在开源软件开发过程中,哪些指标会影响社区活跃度?

Q2:在开源软件开发过程中,哪些指标会影响团队效率?

Q3:在开源软件开发过程中,哪些指标会影响开源软件质量?

针对Q1提出的问题,基于文献[15,16]可以知道行业经常使用Star作为开源软件流行度的度量标准,Github社区本身也基本采用流行度来评判开源软件的受欢迎程度. 因此,为了解决这一问题,首先分析所有采集到的数据特征和Star的相关性,获取与Star强相关的特征集.

针对Q2提出的问题,传统的软件工程领域经常使用团队的关闭问题数(关闭的问题包括新需求、改进请求和缺陷)、人均代码行或代码变更速度作为软件团队绩效的度量标准. 因此,通过对其进行相关性分析,获得影响团队效率的特征集.

针对Q3提出的问题,将所有采集到的数据特征和缺陷数、缺陷关闭数进行相关性分析,获得影响软件质量的特征集.

最终,通过特征间相关性分析和特征物理含义分析,得到如表1所示的属性列表.

表1 特征选择结果特征集汇总列表

编号	属性名称	属性说明	编号	属性名称	属性说明
1	stars	星数	16	code_change_speed	代码变更速度
2	subscribers	关注度,对应Watch	17	defect_num	缺陷数
3	forks	复制数,代表贡献者本地复制代码总数	18	defect_close_num	缺陷关闭数
4	issue_comment_num	问题评论数	19	open_pull	开放合并请求数
5	issue_detail_num	问题数	20	open_issues	开放问题数
6	close_issues	问题关闭数	21	acommit	第1活跃用户的提交数
7	pull_num	合并请求数	22	bcommit	第2活跃用户的提交数
8	pull_comment_num	合并请求评论数	23	ccommit	第3活跃用户的提交数
9	close_pull	合并请求关闭数	24	dcommit	第4活跃用户的提交数
10	pull_merge_num	合并数	25	ecommit	第5活跃用户的提交数
11	mainbranch_commits	主分支代码提交数	26	fcommit	第6活跃用户的提交数
12	active_num	活跃的贡献者数	27	gcommit	第7活跃用户的提交数
13	contributors	贡献者人数	28	hcommit	第8活跃用户的提交数
14	codesize	代码行数	29	icommit	第9活跃用户的提交数
15	codesize_per	人均代码行	30	jcommit	第10活跃用户的提交数

3.2.2 特征构建

特征构建就是从原始数据中人工构建新的特征,以支持开源软件健康状态的全面感知和诊断.针对开源软件开发过程的特征构建,首先提出以下两个问题.

Q4:针对开源软件项目特征,哪些传统的软件工程指标可以引入到开源软件的状态感知过程中?

Q5:针对开源软件项目,哪些指标可以用来对开源软件进行分类?

针对问题Q4,在表1的特征集中,需要重点考虑三个方面的传统软件工程度量指标的应用.首先传统软件工程对软件质量的评价经常使用缺陷率、缺陷关闭率和缺陷关闭速度作为软件质量状态的度量标准.因此,在结果特征集中使用千行误码率(defect_rate)、缺陷关闭率(defect_closing_rate)和缺陷关闭速度(defect_closing_speed)三个派生度量元代替原来的缺陷数和缺陷关闭数,来度量开源软件的软件质量状态.考虑到指标方向性的问题,通过对千行误码率取倒数,生成新的每缺陷代码行指标(rate_defect),方便进一步使用机器学习算法对数据集进行分析;其次,考虑到问题评论数、问题关闭数、主分支代码提交数、合并请求数、合并数、代码行数等属性一方面与项目的生存时间有关,另一方面还与团队规模(contributors)有关,直接在不同的项目之间对比分析上述指标是不恰当的,因此,在选择的特征集中包含了速度型派生度量元和人均型派生度量元;最后,考虑到敏捷开发经常用发布心跳来衡量团队的生产节拍及健康状态,基于开源软件的对外发布数(release_num)和内部标签数(tag_num),构建出开源软件的外部发布心跳(release_speed)和内部心跳(tag_speed).

针对问题Q5,由软件工程实践经验可知,不同的项目规模、变更规模、编程语言和许可证类型等都会对开源软件项目的成功有影响.文献[17]明确指出了不同的许可证类型和编程语言对开源软件项目的成功有直接影响.因此,在选择的特征集中包含了项目规模、变更规模、编程语言和许可证类型.

3.2.3 特征提取

特征提取就是要自动地构建新的特征,将原始特征转换为一组具有明显物理意义或者统计意义的特征.综合考虑特征提取和降维两个方面,可以选择的方法包括奇异值分解(Singular Value Decomposition, SVD)、主成分分析(Principal Component Analysis, PCA)、多维尺度分析(Multidimensional Scaling, MDS)和线性判别分析(Linear Discriminant Analysis, LDA)等,综合考虑数据集的数据样本分布、大小和结果的可解释性等方面,本文选用了主成分分析方法进行特征提取,结果显示前7个主成分的特征值之和占总方差的81.836%,信息损失在可接受的范围内.深入分析主成分分析旋转后的成分矩阵中每个主成分的构成属性的软件工程含义,如表2所示,最终将7个主成分定义为贡献者活跃度、社区活跃度、合并请求活跃度、问题解决活跃度、团队编码效率、团队心跳和软件质量七个量化指标.

表2　开源软件项目健康状况自感知模型

主要维度	一级标签(7个)	二级标签(15个)	特征指标	特征说明
社区视角	Community_Vitality(社区活跃度)	流行度	stars	星数
		关注度	subscribers	关注度,对应Watch
		开发者活跃度	forks	复制数
			issue_comment_num	问题评论数
过程视角	Issue_Vitality(问题解决活跃度)		issue_comment_num_per	人均问题评论数
			close_issues_per	人均问题关闭数
	Pull_Request_Vitality(合并请求活跃度)		pull_comment_num_per	人均合并请求评论数
			close_pull_per	人均合并请求关闭数
			pull_request_speed	团队合并请求速度
		人均合并请求速度	pull_request_speed_per	人均合并请求速度
			merge_speed	团队合并速度
		人均合并速度	merge_speed_per	人均合并速度
团队视角	Team_Heartbeat(团队心跳)	外部发布心跳	release_speed	外部发布心跳
		内部心跳	tag_speed	内部心跳
	Contributor_Vitality(贡献者活跃度)		active_num	活跃的贡献者数
			bcommit	第2个活跃用户的代码提交数
			mainbranch_commits	主分支代码提交数
			close_pull	合并请求关闭数
		变更规模	close_issues	问题关闭数
			pull_comment_num	合并请求评论数
			commit_speed	上一年平均代码提交速度
	Team_Coding_Efficiency(团队编码效率)		codesize_per	人均代码行
			mainbranch_commits_per	人均主分支提交数
		代码变更速度	code_change_speed	代码变更速度
		每缺陷代码行	rate_defect	千行误码率(倒数)
		团队规模	contributors	贡献者人数
软件视角	Software_Quality(软件质量)	缺陷关闭速度	defect_closing_speed	缺陷关闭速度
		缺陷关闭率	defect_closing_rate	缺陷关闭率
		代码规模	codesize	代码行数
		编程语言	langcount	编程语言数量
			mlangsize	主语言代码量
			mlangper	主语言百分比
			slangsize	第二语言代码量
			slangper	第二语言百分比
			tlangsize	第三语言代码量
			tlangper	第三语言百分比

3.3　自感知模型定义

为了方便开发者直观地感知开源软件健康水平,本文构建了一个分层的、可量化的开源软件健康状态自感知模型,如表2所示,它主要由4大视角、7个一级标签、15个二级标签和36个特征指标组成. 其中,4个视角分别是社区视角、过程视角、团队视角和软件视角;7个一级标签基于主成分分析结果聚类生成,15个二级标签通过特征集中的相关特征自动聚类计算生成.

7个一级标签由PCA分析获得的七个主成分聚类后生成. 深入分析主成分分析旋转后的成分矩阵中每个主成分的构成属性的软件工程含义后,分别将它们定义为贡献者活跃度、社区活跃度、合并请求活跃度、问题解决活跃度、团队编码效率、团队心跳和软件质量,每个项目在该主成分上的得分用来表征对应维度上的健康得分. 进一步通过对相应主成分进行聚类计

算,通过一级标签将相关项目分为高中低三类. 并且,通过本研究团队前期已发表的模型计算方法[18],基于7个主成分的分值可以进一步计算出每个项目在四大视角上的健康度得分和开源软件健康状态感知指数(Health Status Perception Index, HSPI).

15个二级标签有两种类型特征指标组成,一种是有明确量值的特征指标,如stars、subscribers等,这些值可以在经过标准化变化后表示项目在该特征指标上的得分;另一种则是分类特征. 通过对这些特征指标进行聚类计算后获得的二级标签结果,例如,团队心跳标签是通过对开源软件项目的外部发布心跳(release_speed)和内部心跳(tag_speed)进行聚类实现的,聚类结果将数据集分为三类,通过分析其软件工程含义后分别定义为内低外低型(对外发布数低、内部标签数也低,占47.3%)、内低外高型(内部标签数较低,但对外发布数较高,占25%)和内高外高(内部标签数和对外发布数都高,占27.7%)三个类别;编程语言标签是通过对开源软件开发项目的所使用的语言数量、主要语言的代码量和百分比进行聚类得到,可以分别归纳为单语言型、单语言主导型(虽有多种语言,但主语言占绝对优势)和多语言型.

至此,基于特征工程和存储库数据挖掘方法,本文定义了一个由4大视角、7个一级标签、15个二级标签和36个特征指标共同组成的项目健康状态量化自感知模型. 考虑到CSS只是开源软件的子集,因此,自感知模型完全适用于CSS的健康状态自感知过程.

3.4 使用自感知模型进行CSS健康状态自感知的应用验证

基于CSS健康状态自感知模型,每个项目都可以基于存储库数据自动生成由7个一级标签和15个二级标签组成的项目画像,帮助用户快速了解开源软件的健康状态. 在宏观上,通过以感知模型中的标签为分类变量进行统计分析对比,可以帮助软件工程研究人员分析开源软件生态系统的统计学分布特征;在微观上,通过开源软件的标签分类结果,可以帮助开源软件使用者一目了然地了解开源软件的健康状态.

以JeeSite开源项目为例,说明CSS自感知模型的应用效果. JeeSite是一个基于多个优秀的开源项目,高度整合封装而成的高性能、强安全性的开源Java EE快速开发平台,是一个典型的CSS. 它定位于企业信息化领域,已内置企业信息化系统的基础功能和高效的代码生成工具,包括:系统权限组件、数据权限组件、数据字典组件、核心工具组件、视图操作组件、工作流组件、代码生成等. JeeSite自2013年发布以来已被广大爱好者用到了企业、政府、医疗、金融、互联网等各个领域中,JeeSite依靠架构简单精良、易于扩展、大众思维的设计模式,得到一致好评,于2016和2017连续两年获得开源中国《最受欢迎中国开源软件》奖杯. 自感知模型标签体系是基于特征工程获取的开源软件的特征集自动聚类生成,基本上反映了开源软件主要特征的分布状态,能够帮助开发者一目了然地了解CSS的健康状况. JeeSite的自感知结果如表3所示.

表3 JeeSite项目自感知结果

主要维度	一级标签/分类标签	感知结果
社区视角	社区活跃度	高
过程视角	问题解决活跃度	中
	合并请求活跃度	中
团队视角	团队心跳	低
	贡献者活跃度	中
	团队编码效率	高
	团队规模	中
软件视角	软件质量	中
	代码规模	高
	编程语言	多语言型

更进一步,通过将标签体系中流行度、团队规模、变更规模、代码规模和编程语言等作为分类变量,分析其变化对其它标签维度和特征平均值分布情况的影响,可以发现开源软件开发生态系统存在以下宏观特征:

(1)大多特征均值会随着流行度的提高而增加,例如问题解决活跃度、合并请求评论数和问题关闭数等,这很好地解释了为什么开源社区很多人直接使用流行度(Star数)来度量开源软件项目的好坏;

(2)流行度高的项目,人均代码行、人均代码变更及缺陷关闭速度等会下降. 这进一步说明了即使在开源软件社会化开发方式中,软件开发过程也是规模不经济的.

(3)虽然变更对于社会化开发的开源软件是好事,是开源软件健康发展的标志之一;但过快过大的软件变更可能会导致开源软件质量下降. 因此,当变更规模变大时,团队应该更加关注软件质量.

(4)团队编码效率和软件质量与团队规模呈现出相反的趋势. 因此,当团队规模变大时,人均的工作效率可能降低,而且软件质量应该受到更多关注.

(5)开源软件团队采用单语言主导型(主语言的比例平均约为90%)有助于项目成功. 这说明开源软件团队采用一种主导的编程语言时,可以降低团队的学习成本、提高沟通效率,有助于项目成功.

(6)随着代码规模的增加,缺陷量会随之增加,而且团队解决缺陷的时间会逐步变长.

4　CSS健康状态自诊断模型定义

自感知模型中的一级标签都是分类标签,它将存储库中的相关软件项目在每个维度上都分成了三个类别,例如,社区活跃度标签,它分为“高”、“中”、“低”三种状态. 如图4所示,以其为分类变量分类统计量化自感知模型中对应主成分的均值,通过将不同维度的均值按“高”、“中”和“低”连在一起分别组成健康线、预警线和问题线. 因此,仿照质量控制图,本文定义了一个能够可视化进行项目健康诊断的项目自诊断质量控制图,其中的上控制线为健康线,下控制线为问题线,中心线为预警线. 这样在项目自诊断控制图中,凡是低于中心线的项目指标,都可以认为进入了预警状态,凡是低于问题线的项目指标,则进入故障状态. 因此,基于七个一级标签维度上的健康线、预警线和问题线,定义出用于开源软件健康诊断的自诊断模型,可以帮助团队自动判断CSS是否存在问题和出问题的维度,方便团队基于存储库数据自动完成项目健康问题的自诊断. 在计算过程中,为了避免极值影响,对所有主成分进行了上下1%的缩尾处理.

图4是以JeeSite开源项目为例的健康状态自诊断结果. 从图中可以看出,JeeSite项目在社区活跃度、团队编码效率等方面都远超健康线,但在团队心跳、合并请求活跃度等方面还有提升的空间,尤其是团队心跳方面,目前低于问题线,为故障状态,需要引起开发团队的注意,及时采取改进措施保证项目的健康快速成长.

针对JeeSite项目团队心跳维度低于预警线的情况,需要进一步查看原因. 因此,可以获得JeeSite项目在团队心跳子维度上的详细结果,如表4所示,可以看到JeeSite项目在外部发布心跳、内部心跳等方面表现较差,均低于问题线,处于故障状态,提醒其开发团队

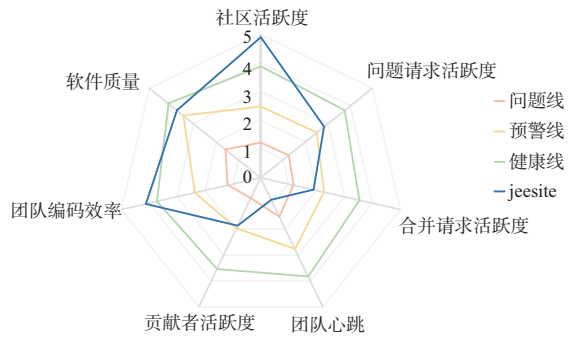

图4　jeesite健康状态诊断图

应尽快制定改进措施,确保按时发布对外版本和内部的里程碑版本.

表4　JeeSite项目的团队心跳详细结果

属性	属性值	状态	问题线	预警线	健康线
外部发布心跳	0.028	故障	0.083	0.141	0.4
内部心跳	0.111	故障	0.288	0.255	0.985

进一步地,JeeSite项目还包含众多成员软件,如前端JS框架为jquery,缓存框架为redis,下拉选择框为select2,数据库连接池为druid,客户端验证为jquery-validation. 其成员软件各维度的健康状态感知如表5所示,通过表5可以看出,select2开源项目和jquery-validation开源项目的HSPI指数偏低,在具体维度上,select2开源项目的软件质量维度、团队编码效率维度上得分较低,该指标以量化地方式明确地告诉JeeSite开发团队必须做好准备,更换掉可能出问题的成员软件;jquery-validation开源项目的团队编码效率和团队心跳维度还有较大的提升空间,该指标以量化地方式明确地告诉该成员软件的开发团队努力方向. 因此,CSS团队通过了解成员软件的健康状态,可以及时采取措施保证CSS软件的健康演化,如对HSPI指数低于一定阈值的成员软件及时寻找替换软件等.

表5　JeeSite成员软件健康状态感知结果

开源项目	社区活跃度	合并请求活跃度	问题解决活跃度	软件质量	贡献者活跃度	团队编码效率	团队心跳	HSPI
jquery	5	2.743	2.993	3.134	4.706	1.868	4.006	3.417
redis	5	2.621	3.296	3.17	4.703	2.490	4.163	3.473
select2	5	2.357	3.268	1.823	4.018	0.229	2.731	2.718
druid	5	3.34	3.245	3.191	4.178	3.361	3.12	3.410
jquery-validation	4.695	2.073	2.989	2.886	3.307	1.437	2.076	2.520

另外,针对时序数据,同样可以采用质量控制图的原理对开源项目进行自诊断. 图5和图6分别展示了tensorflow项目、Tomcat-Research项目在代码变更速度属性上的质量控制图. 从图5中可以看到,在项目初期,tensorflow项目的代码变更速度不稳定,并出现了低于预警线的情况;随着项目的不断成长,代码变更速度逐渐稳定,并远高于健康线.

从图6中可以看到,Tomcat-Research项目的代码变更速度上不够稳定,长期低于健康线,甚至出现了低于问题线的情况. 针对这种情况,可以运用7点运行定律,即如果在一个质量控制图中,一行上的7个数据点都低于平均值或高于平均值,或者都是上升的,或者都

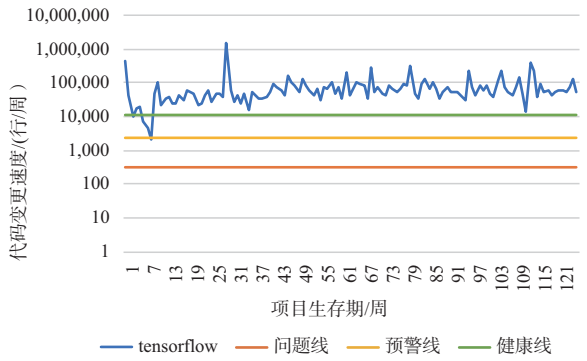

图5 Tensorflow在代码变更速度上的质量控制图

是下降的,那么这个过程就被认定为非随机问题而接受检查. 具体来说,以每周为单位,可以看到Tomcat-Research项目的代码变更速度在第315~321周(以首次代码变更所在周为第1周)出现了连续7个点低于预警线的情况,应该及时报警,引起CSS团队的注意,尽早制定改进措施.

综上分析可知,本文提出的由健康线、预警线和问题线组成的软件自诊断模型,能够帮助复杂软件系统的开发人员快速感知CSS及其所依赖的成员软件的健康状况,并实时做出预警.

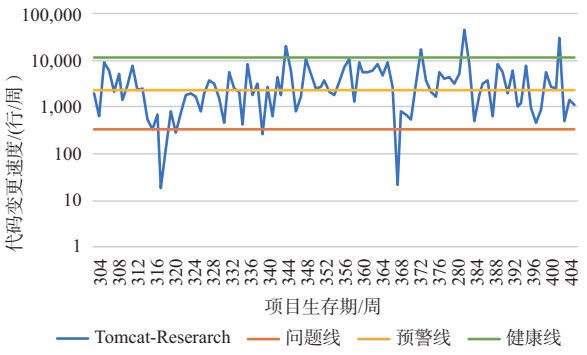

图6 Tomcat-Research项目在代码变更速度上的质量控制图

5 模型对比分析

5.1 社区活跃度和流行度的状态感知效果对比

因为开源社区常用流行度(star)来区分开源软件好坏,因此,首先对比感知模型一级标签"社区活跃度"和流行度对项目好坏的感知效果. 实验中,将流行度和社区活跃度作为分类变量对数值化的各标签取平均值进行对比,如图7所示. 仔细观察对比结果发现,在项目好时,社区活跃度在各个特征维度上均值都较高;而在项目不好时,它在各个特征维度上的均值又都较低,这说明使用本文提出的感知模型的"社区活跃度"标签作为判别项目流行度比现在普遍使用的流行度效果更好.

5.2 不同开源软件健康状态评价方法的横向对比分析

目前,许多研究者在开源软件健康状态评价上进行了深入研究,并提出了一些评价方法. 为了验证本文提出感知模型的有效性,与其中的两个方法进行了对比. 其中,OSSpal方法[19]指出高质量开源软件项目最有效的指标为贡献者人数(contributors)、提交数(main-branch_commits)、订阅者数(subscribers)、复制数(forks)、开放问题数(open_issues);方法2[20]从用户兴趣、开发者兴趣、发展持续性、版本频率(release_speed)、项目活动等五个方面衡量开源项目成功与否;星数(stars)是GitHub社区中项目流行度的重要指标,Github本身使用此指标作为用户兴趣的依据,项目的贡献者数(contributors)作为开发者兴趣的衡量指标;代码变更速度(code_change_speed)在一定程度上反映了项目的开发活力,是发展持续性的重要指标;项目活动指

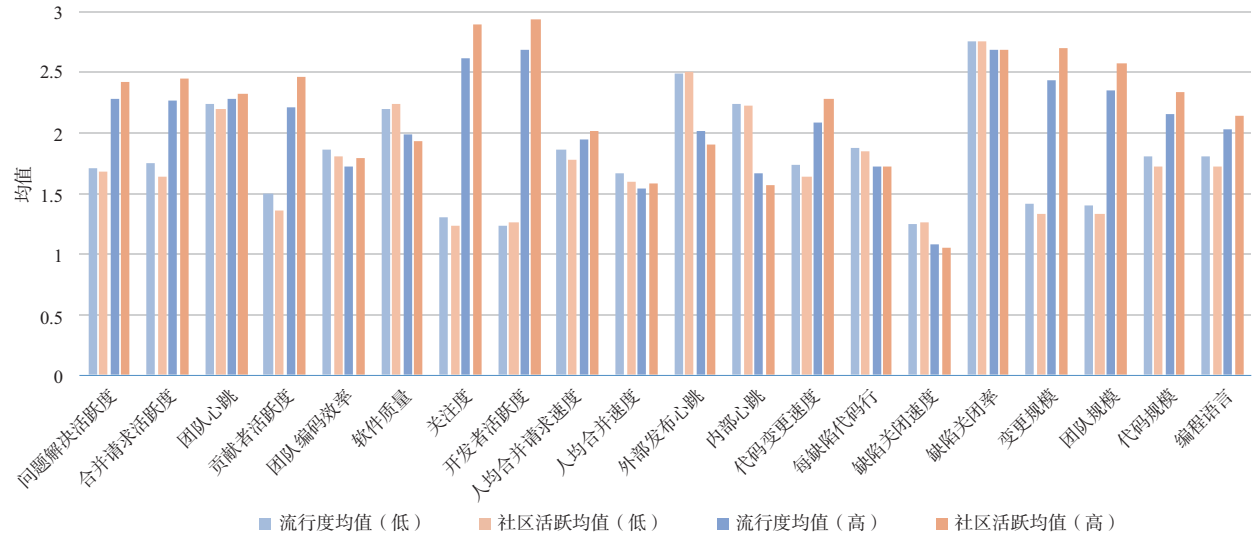

图7流行度和社区活跃度分类对比标签和实际项目特征的均值分布

的是项目开发活动的强度,用主分支代码提交数(main-branch_commits)衡量.

针对本次采样的数据集,对三种评价方法进行对比,结果如图8所示.

Ranking	repository	HSPI
1	kubernetes/kubernetes	1
2	tensorflow/tensorflow	0.95
3	saltstack/salt	0.91
4	moby/moby	0.90
5	elastic/elasticsearch	0.88
6	cockroachdb/cockroach	0.88
7	Microsoft/TypeScript	0.88
8	grpc/grpc	0.86
9	arangodb/arangodb	0.86
10	nodejs/node	0.85

(*a*)本文评价方法结果

Ranking	repository	score
1	moby/moby	1
2	kubernetes/kubernetes	0.93
3	tensorflow/tensorflow	0.88
4	ansible/ansible	0.87
5	golang/go	0.87
6	Microsoft/vscode	0.82
7	DefinitelyTyped/DefinitelyTyped	0.80
8	elastic/elasticsearch	0.79
9	scikit-learn/scikit-learn	0.77
10	angular/angular	0.77

(*b*)方法1(OSSpal)评价结果

Ranking	repository	score
1	kubernetes/kubernetes	1
2	atom/atom	0.90
3	tensorflow/tensorflow	0.84
4	moby/moby	0.82
5	Microsoft/vscode	0.76
6	nodejs/node	0.76
7	golang /go	0.74
8	fastlane /fastlane	0.73
9	mrdoob /three.js	0.70
10	elastic/elasticsearch	0.69

(*c*)方法2 评价结果

图8　模型对比结果

针对上述评价结果进行分析,主要分为以下两种情况:

(1)某些项目出现在三种评价结果的前十名,但在各方法中的具体排名不同,如kubernetes项目、tensorflow项目、moby项目等. Tensorflow项目在方法1的评价结果中排名第三,在方法2的评价结果中排名第三,而在本文的评价结果中排名第二. 同样的,kubernetes在本文方法排名第一,这基本符合实际情况,而moby是一个新的开源项目,它提供了一个组件库、一个组装组件成为容器化的体系框架和一个容器爱好者的社区,其在实际影响力方面不及kubernetes和tensorflow,这充分说明了本方法的优势. 本方法的优势,本质上的原因是由于其综合地考虑了社区、团队、过程质量和软件质量等因素,能够更加全面的评价开源软件,项目排名结果更能真实地反映实际情况.

(2)某些项目部分出现在其他两种评价方法中,在本文评价方法中排名前列,如node项目等. 以node项目为例,在方法2评价结果中排名第六,但却在方法1度量上未进入前十名,经过对该项目的分析发现,该项目是一个让JavaScript运行在服务端的开发平台,其提交数、开放问题数对应指标相对较低,这说明软件本身的质量和发展速度存在隐患,因此未能在方法1中进入前十. 但node项目在其他方面一直较为活跃,因此,在本文评价结果中成为排名靠前的优质项目。由此可见,本文方法由于综合了团队活力、软件质量、过程质量和社区活跃度等度量维度,能够更加全面的评价开源软件.

通过三种评价结果的对比分析,可以看出本文提出的感知模型能够更加全面的感知和度量开源软件,充分验证了该模型的优越性和有效性.

6　总结与展望

面向日益增长的CSS及开源软件的使用规模,如何自动地感知和诊断CSS的健康状态对CSS开发团队至关重要. 为了解决这一问题,本文首先使用特征工程、存储库数据挖掘技术,定义了一个能够全面感知CSS团队状态、软件质量状态、社区状态和软件过程状态的CSS健康状态自感知模型;其次,仿照传统的控制图原理,基于自感知模型一级标签对量化指标的分类统计,本文定义了能够辅助故障诊断的CSS健康状态自诊断模型. 最后,通过对比实验和实证研究很好的验证了模型的有效性和正确性. 不足之处在于由于数据集本身的限制,在当前的感知模型中没有考虑到团队协同平台本身成熟度对CSS健康状态的影响. 未来我们将进一步基于开源软件健康感知和诊断模型,开展CSS智能开发技术和智能推荐技术的研究.

参 考 文 献

[1] 王怀民,吴文峻,毛新军,等.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.
Wang H M, Wu W J, Mao X J, et al. Growth structure and adaptive evolution of complex software systems [J]. Chinese Science: Information Science, 2014, 44 (6): 743 – 761. (in Chinese)

[2] 中国信息物理系统发展论坛. 信息物理系统白皮书. [DB/OL]. http://www.cesi.cn/201703/2251.html, 2018-03-01.

[3] 阿里研究院."互联网+"——中国经济新引擎[DB/OL]. http://www.aliresearch.com/blog/article/detail/id/20324.html, 2017-03-24.

[4] Lemos R D, Giese H, Müller H A, et al. Software engineering for self-adaptive systems: a second research roadmap [A]. Software Engineering for Self-Adaptive Systems II [C]. Berlin Heidelberg: Springer, 2013. 1 – 32.

[5] Ralph P . Toward methodological guidelines for process theories and taxonomies in software engineering[J]. IEEE

Transactions on Software Engineering, 2019, 45(7): 712 − 735.

[6] Shevtsov S , Berekmeri M , Weyns D , et al. Control- theoretical software adaptation: a systematic literature review [J]. IEEE Transactions on Software Engineering, 2018, 44 (8):784 − 810.

[7] IBM Corporation. An Architectural Blueprint for Autonomic Computing. [EB/OL] http://www-03.ibm.com/autonomic/pdfs/AC_Blueprint_White_Paper_4th. pdf, 2017 - 06 -24.

[8] Dobson S, Denazis S G, Fernández Antonio, et al. A survey of autonomic communications [J]. ACM Transactions on Autonomous and Adaptive Systems, 2006, 1(2): 223 − 259.

[9] Graziotin D , Lenberg P , Feldt R , et al. Psychometrics in Behavioral Software Engineering: a Methodological Introduction with Guidelines[EB/OL]. https://arxiv. org/pdf/ 2005.09959.pdf, 2020-10-19

[10] Crowston K, Howison J, Annabi H. Information systems success in free and open source software development: theory and measures[J]. Software Process Improvement & Practice, 2010, 11(2):123 − 148.

[11] Margan D, Čandrlić S. The success of open source software: A review[A]. 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) [C]. Piscataway, NJ: IEEE, 2015. 1463 − 1468.

[12] Raja U, Tretter M J. Defining and evaluating a measure of open source project survivability[J]. IEEE Transactions on Software Engineering, 2012, 38(1):163 − 174.

[13] Borges H , Tulio Valente M . What's in a GitHub star? understanding repository starring practices in a social coding platform[J]. Journal of Systems and Software, 2018, 146:112 − 129.

[14] Jacques V. Introduction of Pygithub[DB/OL]. http://pygithub.readthedocs.io/en/latest/introduction.html, 2017-06 -09.

[15] Hu Y , Wang S S, Ren Y Z, et al. User influence analysis for Github developer social networks[J]. Expert Systems with Applications, 2018, 108:108 − 118.

[16] Borges H , Hora A , Valente M T . Understanding the factors that impact the popularity of GitHub repositories[A]. 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME) [C]. Raleigh, NC, USA: IEEE, 2016. 334 − 344.

[17] 张得光,李兵,何鹏,等. 基于软件生态系统的开源社区特性研究[J]. 计算机工程, 2015, 41(11):106 − 113.
Zhang D G, Li B, He P, et al. Characteristic study of open source community based on software ecosystem [J]. Computer Engineering, 2015, 41 (11): 106 − 113. (in Chinese)

[18] 宁德军, 叶培根, 刘琴,等. 基于存储库数据挖掘的开源软件成功度量方法[J]. 电子学报, 2018, 46(012):2930 − 2935.
Ning D J, Ye P G, Liu Q, et al. Open source software success measurement method based on mining software repository [J]. Acta Electronica Sinica, 2018, 46 (12): 2930 − 2935. (in Chinese)

[19] Wasserman A I , Guo X , Mcmillian B , et al. OSSpal: finding and evaluating open source software [A]. IFIP International Conference on Open Source Systems [C]. Buenos Aires, Argentina: Springer, Cham, 2017. 193 − 203.

[20] Ghapanchi A H . Investigating the interrelationships among success measures of open source software projects [J]. Journal of Organizational Computing and Electronic Commerce, 2015, 25(1):28 − 46.

作者简介

王 森 男,1966 年 8 月出生于上海. 教授级高工,主要研究方向为工业互联网、智能制造.
E-mail:wangsen@baosight.com

王 煜 女, 1994 年 1 月出生于山东临沂. 研究生,主要研究方向为软工数据智能、数据挖掘.
E-mail:wangyu02@sari.ac.cn

宁德军(通信作者) 男,1972 年 7 月出生于黑龙江省. 教授级高级工程师,CCF 会员. 长期从事下一代软件工程技术、大数据智能技术和海云协同计算研究.
E-mail:ningdj@ sari.ac.cn