软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(11):3380−3403 [doi: 10.13328/j.cnki.jos.005830]

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

软件需求变更管理的系统动力学仿真建模*

康燕妮[1], 张 璇[1,2], 王 旭[3], 李 彤[1,2], 唐子淇[1], 牛家梅[1]

[1](云南大学 软件学院,云南 昆明 650091)

[2](云南省软件工程重点实验室(云南大学),云南 昆明 650091)

[3](云南大学 经济学院,云南 昆明 650091)

通讯作者: 张璇, E-mail: zhxuan@ynu.edu.cn

摘 要: 软件需求变更频繁发生,给软件项目造成了诸多威胁.能否对需求变更进行有效的控制管理,决定着软件的成败.使用系统动力学方法对软件需求变更管理过程进行仿真建模,可以动态地分析并预测需求变更产生的原因以及变更对软件项目造成的影响;对软件需求变更管理过程改进进行系统动力学仿真,亦可以辅助软件项目组织选择合适的过程改进策略.因此,基于系统动力学方法,参考了敏捷过程进行开源软件需求变更管理过程的建模和模型检测.以 Spring Framework 项目为研究案例,进行该项目 3.2.x 分支的软件需求变更管理过程的系统动力学仿真分析,并对需求变更管理进行过程改进仿真.通过对过程改进的仿真结果进行比对,说明各改进策略均降低了基线数据的软件缺陷率,提高了软件质量.根据软件项目的成本和进度要求,给出了过程改进建议.

关键词: 系统动力学;敏捷过程;开源软件;软件过程仿真;软件过程改进

中图法分类号: TP311

中文引用格式: 康燕妮,张璇,王旭,李彤,唐子淇,牛家梅.软件需求变更管理的系统动力学仿真建模.软件学报,2020,31(11):3380−3403. http://www.jos.org.cn/1000-9825/5830.htm

英文引用格式: Kang YN, Zhang X, Wang X, Li T, Tang ZQ, Niu JM. System dynamics simulation modeling of software requirements change management. Ruan Jian Xue Bao/Journal of Software, 2020,31(11):3380−3403 (in Chinese). http://www.jos.org.cn/1000-9825/5830.htm

System Dynamics Simulation Modeling of Software Requirements Change Management

KANG Yan-Ni[1], ZHANG Xuan[1,2], WANG Xu[3], LI Tong[1,2], TANG Zi-Qi[1], NIU Jia-Mei[1]

[1](School of Software, Yunnan University, Kunming 650091, China)

[2](Yunnan Provincial Key Laboratory of Software Engineering (Yunnan University), Kunming 650091, China)

[3](School of Economics, Yunnan University, Kunming 650091, China)

Abstract: Software requirements change frequently, which pose many threats to software projects. Effective management of requirements change determines the success or failure of the software project. System dynamics can be used to simulate the process of software requirements change management, aiming to dynamically analyze and predict the cause of requirements change and the effects of change on software projects. System dynamics also can assist software organizations to improve requirement change management

* 基金项目: 国家自然科学基金(61862063, 61502413, 61262025, 61379032, 61662085); 国家社会科学基金(18BJL104); 云南省科技计划(2016FB106); 云南省软件工程重点实验室开放基金(2015SE202); 云南省创新团队“数据驱动的软件工程创新团队”项目(2017HC012)

Foundation item: National Natural Science Foundation of China (61862063, 61502413, 61262025, 61379032, 61662085); National Social Science Foundation of China (18BJL104); Science and Technology Plan of Yunnan Province (2016FB106); Open Fund of Yunnan Provincial Key Laboratory of Software Engineering (2015SE202); Program of Innovative Research Team for Data Driven Software Engineering of Yunnan Province (2017HC012)

收稿时间: 2018-05-09; 修改时间: 2018-09-18, 2018-11-08, 2019-01-12; 采用时间: 2019-02-28; jos 在线出版时间: 2019-11-06

CNKI 网络优先出版: 2019-11-06 11:49:07, http://kns.cnki.net/kcms/detail/11.2560.TP.20191106.1148.003.html

processes. In this study, the system dynamics method is first used to model the process of open source software requirements change management which refers to the agile processes. Then, the models are tested to find out the errors and correct them. Next, taking the Spring Framework as an empirical case study, the system dynamics simulation of the software requirement changes management process of the project 3.2.x branch is carried out. According to the simulation results, the improvement of the requirement change management processes is simulated. By comparing the baseline simulation results with the improvement simulation results, it shows that all the improvements effectively reduce the software defect rate and improve the software quality. In addition, based on the cost and schedule of the software project, the process improvement suggestions are provided.

Key words: system dynamics; agile process; open source software; software process simulation; software process improvement

在当前软件需求急剧增长和迅速变化的时代背景下,要求用户一次性提出所有需求且不再改变是不现实的,可以说,软件需求变更的发生不可避免.然而,对于软件行业的大多数组织来说,在软件生产过程中,软件需求变更的频繁发生使项目组面临重大的困难[1],需求变更给软件的进度、质量及成本等因素造成威胁[2−4],如不能对其进行有效管理,可能会给软件带来诸多负面影响,甚至造成软件失败.因此,能否对软件需求变更进行有效管理,是关乎软件成功与否的重要因素之一.

随着软件需求变更管理研究的逐步深入,使用系统动力学方法(system dynamics,简称 SD)对软件需求变更管理过程进行研究的优势逐渐显现出来.使用系统动力学方法将软件需求变更管理过程视为具有多阶反馈回路的系统,从全局角度,通过因果反馈机制动态分析并预测软件需求变更产生的原因以及变更对软件进度、成本以及质量造成的影响,能够帮助软件项目团队进行有效的需求变更管理,降低需求变更给软件项目带来的风险,最终可以为各软件组织节约可观的时间和成本[5].另外,使用系统动力学方法还可以仿真软件需求变更管理过程改进.基于软件需求变更管理过程所建立的系统动力学仿真模型,是现实软件需求变更管理过程的抽象,通过调整模型中的变量和方程,可以模拟现实中软件需求变更管理过程改进的一系列策略,并根据模型所得的仿真结果来分析改进效果,以达到辅助软件组织进行需求变更管理过程改进的目的.故本文使用系统动力学方法对需求变更管理过程进行建模分析,并对需求变更管理过程改进进行仿真.

当前,大多数学者基于传统软件过程模型进行软件需求变更管理系统动力学仿真研究,基于敏捷过程进行仿真研究相对缺乏[6],而采用敏捷方法进行软件需求变更管理,可以快速、灵活且有效地进行软件需求变更管理,因而是仿真研究领域一个有价值的新研究方向[7].正是基于此,本文参考了敏捷过程模型作为软件需求变更管理过程的系统动力学建模基础.另外,由于开源软件面向公众开放其开发环境[8],对开源软件进行研究可以将研究成果移植到软件组织项目以帮助提升软件质量[9].因此,本文按照图 1 所示的 7 个步骤,基于可获取的开源软件项目数据,参考敏捷过程,进行软件需求变更管理过程的系统动力学仿真建模研究.

- 首先,收集整理系统动力学在软件过程和软件需求变更领域的相关研究文献,对文献进行总结,为后续研究工作提供基础支持.
- 其次,确定模型的范围以及模型需要分析的行为,抽取并标识开源软件过程关键因素,并据标识出因素之间的因果关系,绘制系统动力学因果图.
- 并基于因果图对元素建立的定量和定性描述,建立定量的系统动力学模型,即存量-流量图.
- 在明确模型变量后,定义速率方程,并设置所选变量的初始值,对所建模型进行系统动力学检测,纠正发现的模型错误.
- 在模型建立以后,以开源 Spring Framework 项目为研究案例,对其版本分支 3.2.x 的需求变更管理过程进行仿真和分析,建立过程改进基线.
- 最后进行过程改进仿真,并对仿真结果与基线结果进行对比.根据对比分析,给出需求变更项目管理过程的改进建议.

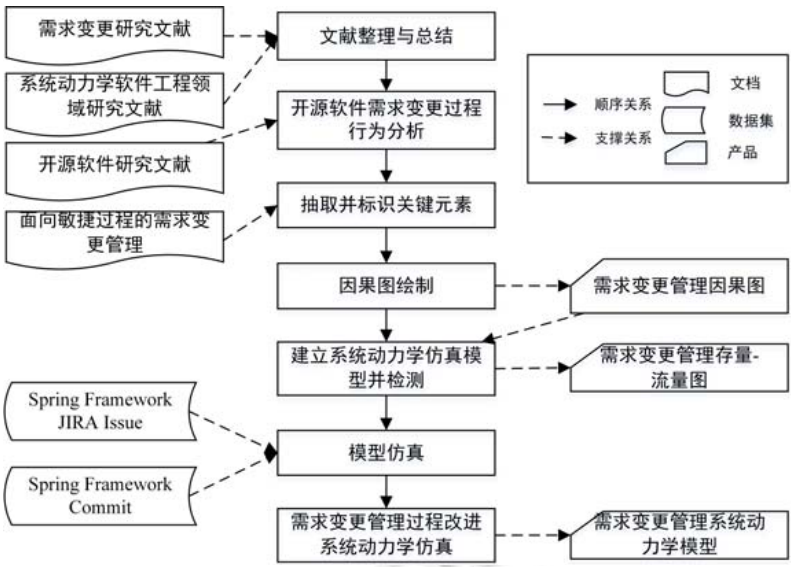


Fig.1 Research framework of system dynamics simulation modeling for requirements change management

图 1 需求变更管理的系统动力学仿真建模研究框架

1 开源软件需求变更管理的系统动力学仿真建模

1.1 开源软件需求变更管理行为分析

开源软件的突飞猛进,给软件需求变更管理带来新的挑战.与传统商业软件不同,开源软件的需求量更为庞大且不断变化,这就导致需求变更发生更为频繁.通常情况下,开源软件的需求变更管理一般采用轻量级和灵活的工具和方法来辅助完成.Issue tracking system 作为当前用于开源软件需求管理最为流行的工具之一,可以对需求变更请求进行有效的收集、评审以及跟踪.另外,软件工程领域,敏捷方法所具有的快速响应、不断迭代的特点与开源软件需求变更管理的要求相符,且根据已有调研,当前还没有使用系统动力学方法对开源软件需求变更管理过程进行仿真建模的文献发表.因此,本文参考 Scrum 敏捷方法,对使用 issue tracking system 的开源软件需求变更管理过程进行系统动力学仿真建模研究.

根据文献[10]和相关开源软件需求变更过程文献[11]的分析,一个典型的开源软件需求变更管理过程主要由 7 个步骤组成,如图 2 所示.

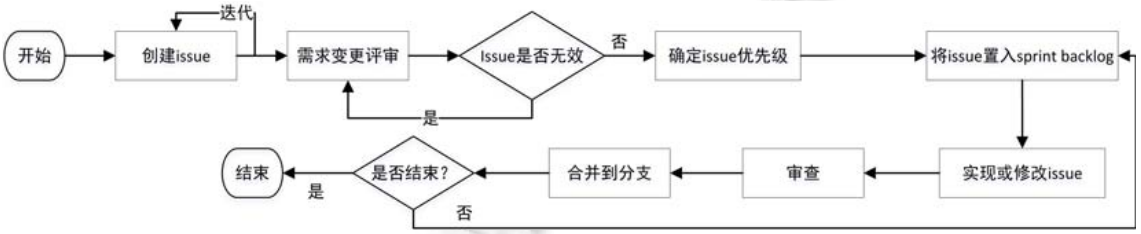


Fig.2 Requirements change management process for open source software

图 2 开源软件需求变更管理过程

整个过程分别由需求变更请求创建(即 issue 创建)、分类 issue、确定 issue 优先级、选取 issue 到 sprint 中进行迭代开发、实现或者修改 issue、审查 issue 变更相关代码、使用 commit(提交)命令把实现或修改 issue 而发生变更的代码等内容提交到对应分支并不断进行 sprint 迭代这 7 个步骤组成.

需要说明的是,在 issue tracking system 中,需求变更请求以及系统中的缺陷都统称为 issue.本文将需求变更请求与系统中的缺陷加以区分,本文中的 issue 专指需求变更类型的 issue,而系统的缺陷类的 issue 则称为 bug.

1.2 开源软件需求变更过程关键因素抽取及因果关系分析

根据上述分析及相关文献整理,选取软件需求变更相关关键因素后,确定各关键因素之间的因果关系,绘制如图 3 所示的因果关系图.因果关系分析起始于需求变更请求(requirement change request),由于需求变更的影响通常直接反映在项目工作量的变化上,为了能够较为详细地分析需求变更对软件项目造成的影响,因果图中将需求变更引起的工作量(requirement change effort)变化细分为 3 类:对需求变更请求分类所需的工作量(triage effort)、实现和修改需求变更请求所需的工作量(implement & modify effort)以及修复 bug 所需的工作量(fix effort).Triage effort 指的是需求变更请求提交后,项目组的成员需要对需求变更请求进行分类(triage),以区分出请求的合理性和有效性所需的工作量,也就是传统软件项目中需求变更评审活动所花费的工作量.目前,大中型开源软件项目通常都使用需求变更管理工具或 issue tracking system 进行软件需求变更管理,这些工具可以在一定程度上帮助项目管理人员进行请求的初步过滤[7],这也是因果图中自动识别有效性(automatic recognition effectiveness)的作用.当需求变更请求被接受,在花费人力和时间去实现或修复这个变更时,就形成了因果图中实现或修改变更请求所需的工作量.另外,在完成需求变更过程中可能会造成 bug 的产生,为了修复这些 bug,相关工作量是修复 bug 所需的工作量(fix effort).

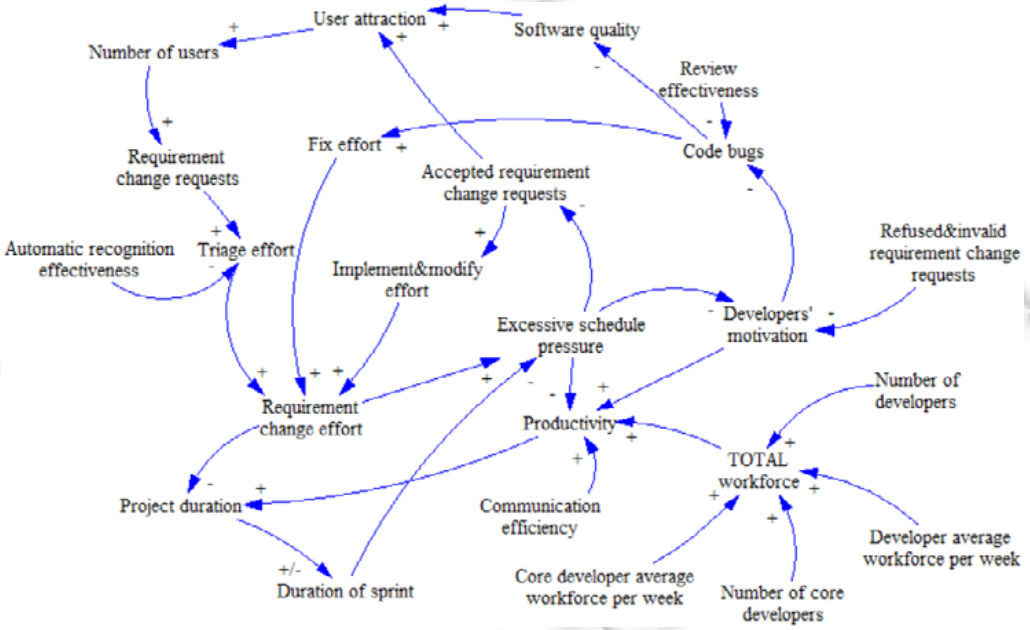


Fig.3 Cause graph for open source software requirement change

图 3 开源软件需求变更因果关系图

随着需求变更请求的增加,需求变更所需消耗的工作量也随之增加,并进一步引起项目持续时间(project duration)的增加以及项目进度压力(excessive schedule pressure)的变化.而项目持续时间的增加,则需要对敏捷过程中每一个 sprint 的迭代周期(duration of sprint)的长短进行调整,以适应完成需求变更所需的工作量要求,从而达到减轻项目压力的目的.项目进度压力作为项目开发过程中的重要影响因素,受 sprint 的迭代周期和完成需求变更所需工作量的共同影响,通过两者的共同控制,把项目进度压力控制在合理有效的范围内,从而保证软件的较低缺陷量(code bugs)以及人员的较高生产力(productivity).另外,软件的低缺陷量是软件高质量(software quality)的特征之一,当软件的质量较高时,能够从一定程度上提升项目对用户的吸引力(user attraction),从而最

终达到对开源软件项目需求变更进行有效管理的目的.上述因果关系图中的各个关键因素是通过综合相关文献以及软件需求变更过程的分析而抽取出来的,这些关键因素通常是产生被观察到行为的重要对象或变量,且一般存在相互促进或抑制的因果关系.

1.3 开源软件需求变更管理系统动力学模型

根据上述对开源软件需求变更过程及因果关系的分析,我们将开源软件的需求变更管理过程系统动力学模型设计为由需求变更实现子系统、开源软件社区人力资源子系统、质量保证子系统、进度计划与控制子系统以及需求变更管理子系统构成的模型,如图 4 所示.

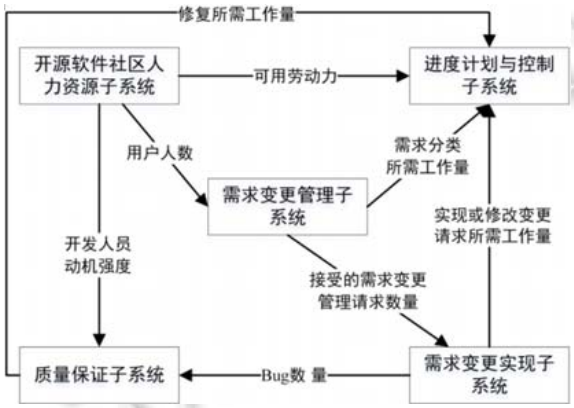


Fig.4 Design of model subsystems

图 4 模型子系统设计

模型中的各个子系统使用系统动力学的存量流量图来进行描述.存量流量图基于因果关系图绘制,明确表示系统的物质流、信息流和反馈作用.在绘制存量流量图时,需整理有关系统的所有因素以及因素之间的相互关系,明确各个存量的意义.通过更进一步地细化因果关系图,并增加必要的补充和完善,保证既能完整地显示出系统应有的因果关系和各模块的正确衔接结构,又能正确反映系统中诸要素的数学意义和数量关系.另外,在建立存量流量图时,应遵循如下一般原则.

1) 每一个反馈回路都至少有一个存量.
2) 只有流量能够改变存量.
3) 一般情况下,存量为系统提供信息,而这信息会用于改变速率变量,表示根据系统状态进行决策,对系统进行控制.
4) 辅助变量都放在信息流中.

基于上述子系统分解及建模原则,下面对各个子系统进行存量流量图建模.

1.3.1 开源软件社区人力资源子系统

基于开源社区人员“洋葱”模型组织方式[12],开源软件社区人力资源子系统(如图 5 所示)的功能是对开源社区的人力资源进行统一配置和管理,使之能够尽可能满足进度计划与控制子系统对人力的要求.

图 6 所示的“洋葱”模型表达了开源软件用户在一定条件下,通过对项目做出贡献后可以成为开发者(developer),而当开发者的贡献度达到要求时或者在某个方面表现突出而满足开源软件项目人员转化条件的情况下,又可以转化成为项目的核心团队成员.采用该模型的开源软件社区的人力由外围的开发人员和核心团队成员共同组成,通常情况下,核心团队成员付出的工作量比非核心团队成员的工作量多出几倍甚至到几千倍,因此,合理地对工作量进行分配,尽可能地调动外围开发者的积极主动性等,相关开源软件社区人力资源管理的重要活动也包含到该子系统的模型中.

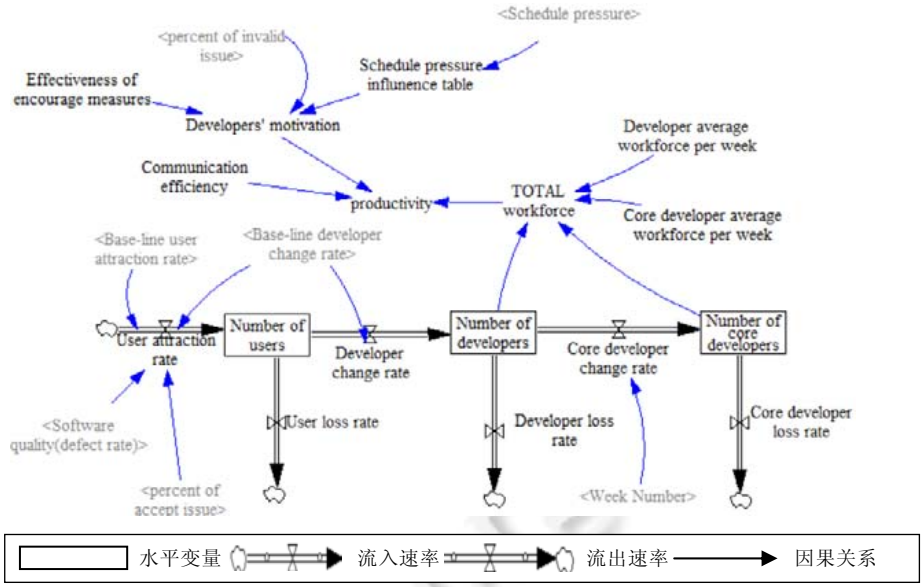


Fig.5 Open source software community human resource subsystem

图 5 开源软件社区人力资源子系统

Fig.6 “Onion” model for open source software development organizations

图 6 开源软件开发组织“洋葱”模型

图 5 给出了开源软件社区人力资源子系统的系统动力学模型.该模型中,各变量的中英文对照、变量释义、单位、方程列举如下(由于篇幅有限,此处只选取关键变量进行解释).

在该子系统中,开发人员动机强度(developers’ motivation)表示的是开发人员士气,受进度压力、激励动机策略有效性以及无效的需求变更请求影响,单位为 Dmnl,计算公式如(1)所示.

$$\left.\begin{aligned} Developers'\ motivation = (1+Schedule\ pressure\ influence\ table)\times(1-percent\ of\ invalid\ issue)\times \\ (1+Effectiveness\ of\ encourage\ measures) \end{aligned}\right\} \quad (1)$$

生产力表示开发人员每周的生产力,即开发人员每周可以解决的 issue 量,单位为 issue/week,计算公式如(2)所示.

$$Productivity=TOTAL\ workforce\times((Developers'\ motivation+Communication\ efficiency)/2) \quad (2)$$

开发人员总劳力(TOTAL workforce)表示外围开发人员和核心开发人员每周可提供的总劳力,单位为 issue/week,计算公式如(3)所示.

$$\left.\begin{aligned} TOTAL\ workforce = Core\ developer\ average\ workforce\ per\ week\times Number\ of\ core\ developers+ \\ Developer\ average\ workforce\ per\ week\times Number\ of\ developers \end{aligned}\right\} \quad (3)$$

开发者人数(number of developers)表示项目外围开发人员总数,单位为 person,计算公式如(4)所示.

$$Number\ of\ developers=\mathrm{INTEG}(Developer\ change\ rate-Developer\ loss\ rate-Core\ developer\ change\ rate,0) \quad (4)$$

INTEG(integration)函数的数学意义是积分,是围绕状态变量建立的.每个状态变量的方程式都是一个积分

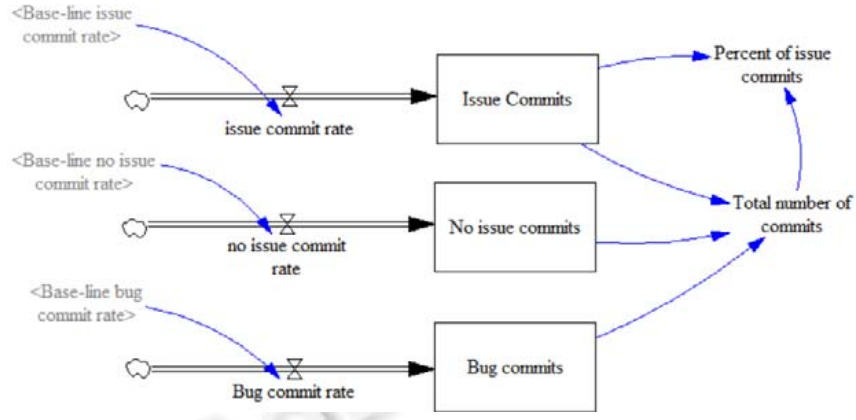
方程.

核心开发人员数(number of core developers)表示项目核心开发人员总数,单位为 person,计算公式如(5)所示.

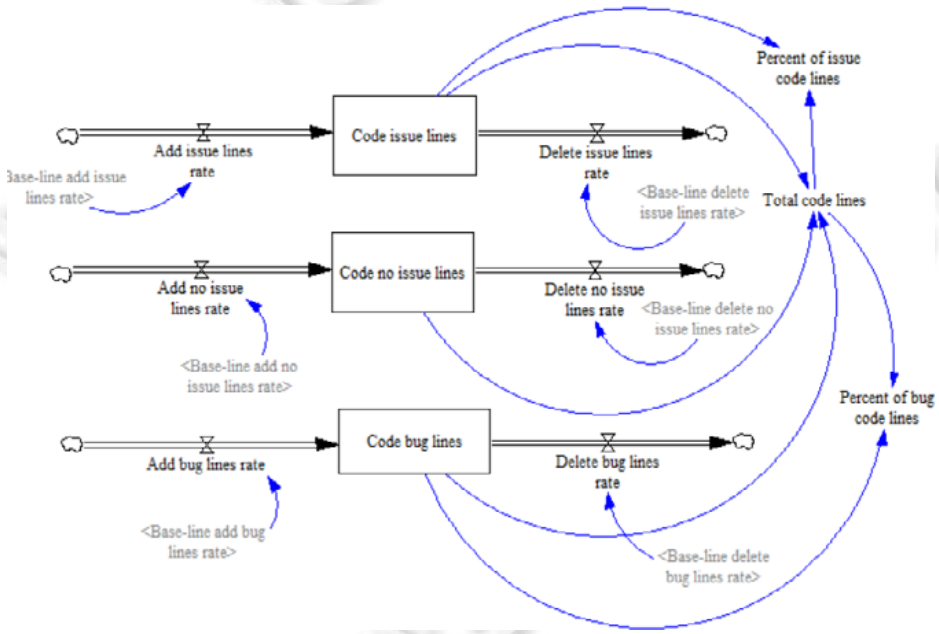
$$Number\ of\ core\ developers=\mathrm{INTEG}(Core\ developer\ change\ rate-Core\ developer\ loss\ rate,0) \tag{5}$$

1.3.2 需求变更实现子系统

需求变更实现子系统的功能是对接受的需求变更请求进行实现和代码修改,该子系统反映开源软件过程中与软件需求变更直接相关的 Commit 提交过程以及代码的变更过程,如图 7 所示.



(a) 提交过程仿真



(b) 编码过程仿真

Fig.7 Requirement change implementation subsystem

图 7 需求变更实现子系统

当一个 issue 被实现或者修改完成之后,要通过 git 版本管理软件中的 commit 命令提交到远端仓库(如图 7(a)所示),而执行一次 commit 命令,被提交的内容包括:为了实现这个 issue 而发生变更的文件以及每个文件增加和删除的代码行数.因此,执行一次 commit 命令就造成代码行数的变更,因此对代码行变更的过程进行建模

(如图 7(b)所示),以分析需求变更对代码行造成的影响,即对软件规模大小造成的影响.由于 commit 提交的内容并不都与 issue 有关,因此,为了更好地分析需求变更对 commit 和代码行的影响,模型中将 commit 提交的有关内容划分为与需求变更有关的 commit 提交(issue commits)、与需求变更无关的 commit 提交(no issue commits)以及与 bug 有关的 commit 提交(bug commits).同样,将与代码行变更有关内容划分为与需求变更有关的代码行变化(code issue lines)、需求变更无关的代码行变化(code no issue lines)以及与 bug 有关的代码行变化(code bug lines)这 3 类.

在该子系统中,需求变更代码总行数(code issue lines)表示与需求变更有关的代码总行数,单位为 line,计算公式如(6)所示.

$$Code\ issue\ lines=\mathrm{INTEG}(Add\ issue\ lines\ rate-Delete\ issue\ lines\ rate,0) \tag{6}$$

项目代码总行数(total code lines)是需求变更有关的代码行数、与需求变更无关的代码行数以及与 bug 代码有关的代码行数之和,单位为 line,计算公式如(7)所示.

$$Total\ code\ lines=Code\ issue\ lines+Code\ no\ issue\ lines+Code\ bug\ lines \tag{7}$$

与需求变更有关的代码百分比(percent of issue code lines),单位为 Dmnl,计算公式如(8)所示.

$$Percent\ of\ issue\ code\ lines=\mathrm{IF\ THEN\ ELSE}(Total\ code\ lines\neq 0,Code\ issue\ lines/Total\ code\ lines,0) \tag{8}$$

1.3.3 质量管理子系统

软件质量是软件成功与否的重要标志.需求变更使得软件的代码发生变更,代码的变更有可能导致 bug 产生,因而影响软件的质量.作为重点分析的目标,软件质量管理单独设计为一个子系统,详细反映需求变更对软件质量带来的影响,如图 8 所示.

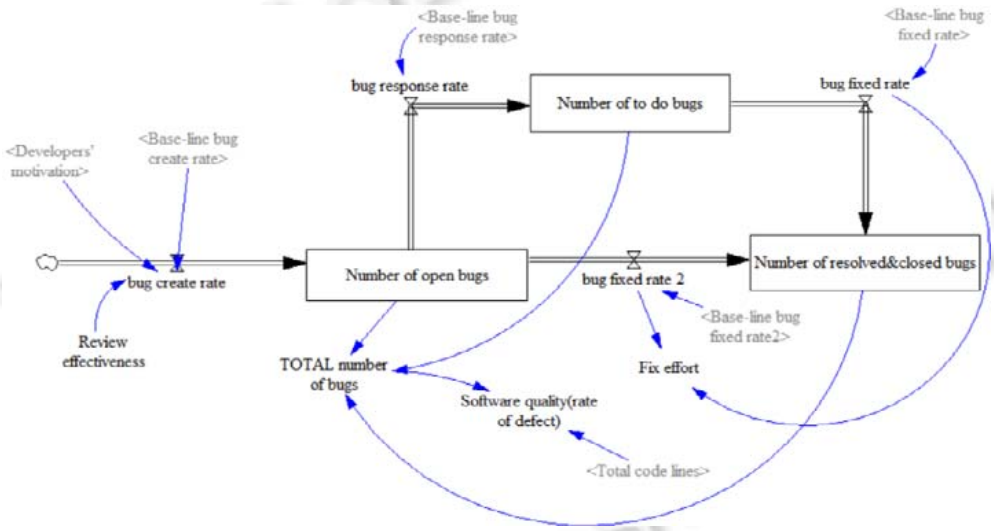


Fig.8 Quality assurance subsystem

图 8 质量保证子系统

软件质量用软件缺陷率来表示,等于 bug 总数量除以代码总行数,计算公式如(9)所示.

$$\left.\begin{aligned}&\text{"}Software\ quality(defect\ rate)\text{"}=\\&\mathrm{IF\ THEN\ ELSE}(Total\ code\ lines\neq 0,TOTAL\ number\ of\ bugs/Total\ code\ lines\times 1000,0)\end{aligned}\right\} \tag{9}$$

评审有效性(review effectiveness)表示对需求变更进行评审,可以减少 bug 报告率,单位为 Dmnl,计算公式如(10)所示.

$$Review\ effectiveness=RANDOM\ NORMAL(0,1,0.05,0.1,0) \tag{10}$$

1.3.4 进度计划与控制子系统

进度计划与控制子系统的功能是对需求变更引起的工作量进行统计,并根据项目的进度压力以及人员的

生产力等因素对项目的持续时间以及 sprint 周期进行计划和控制,以保证项目的进度能够按照计划执行,如图 9 所示.根据第 1.2 节对因果关系的分析,过大的进度压力会造成软件项目的缺陷增加,使软件项目的质量下降,人员的生产率下降以及减少对需求变更请求的接受等负面影响,因此,该系统还对进度压力从调节 sprint 迭代周期以及工作量两个方面进行控制,从而降低过度的进度压力对项目带来的负面影响.

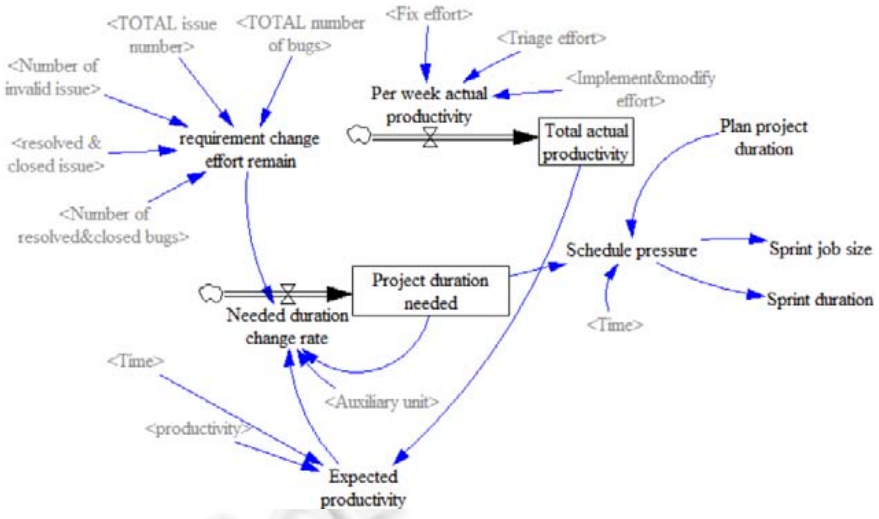


Fig.9 Schedule planning and control subsystem

图 9 进度计划与控制子系统

在该子系统中,进度压力(schedule pressure)是所需项目持续时间除以项目剩余时间,计算公式如(11)所示.

$$Schedule\ pressure=Project\ duration\ needed/(Plan\ project\ duration-time) \tag{11}$$

所需项目持续时间(project duration needed)的单位为 week,计算公式如(12)所示.

$$Project\ duration\ needed=\mathrm{INTEG}(Needed\ duration\ change\ rate,0) \tag{12}$$

每周实际生产力(per week actual productivity)等于所有开发人员每周实际完成的需求变更请求数量、修复 bug 数量以及评审的需求变更请求数量总和,单位为 issue/week,计算公式如(13)所示.

$$Per\ week\ actual\ productivity=Fix\ effort+\text{“}Implement\ \&\ modify\ effort\text{”}+Triage\ effort \tag{13}$$

总的实际生产力(total actual productivity)等于所有开发人员实际完成的需求变更请求数量、修复 bug 数量以及评审的需求变更请求数量总和,单位为 issue,计算公式如(14)所示.

$$Total\ actual\ productivity=\mathrm{INTEG}(Per\ week\ actual\ productivity,0) \tag{14}$$

所需项目持续时间的变化率(needed duration change rate)表示每周随待完成工作量以及实际生产力而变化的项目持续时间,计算公式如(15)所示.

$$Needed\ duration\ change\ rate=\mathrm{IF\ THEN\ ELSE}\left(Expected\ productivity\neq 0,\left(\begin{matrix}requirement\ change\ effort\ remain/\\Expected\ productivity-\\Project\ duration\ needed\end{matrix}\right)\Big/Auxiliary\ unit,0\right) \tag{15}$$

1.3.5 需求变更管理子系统

需求变更管理子系统发挥着需求变更管理的功能,作为模型的核心部分,参考 Scrum 敏捷过程进行建模.Scrum 敏捷过程能够快速地对需求变更做出响应,并通过优先权评估,工作量评估等一系列的评估活动,以及不同的管理策略对需求变更进行有效地管理.该子系统如图 10 所示.

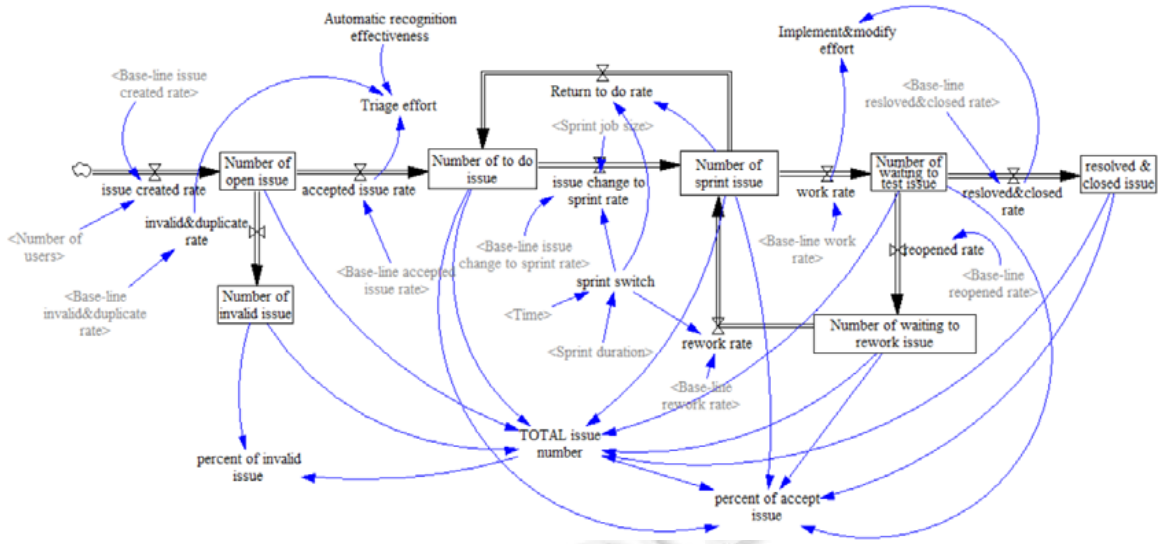


Fig.10 Requirements change management subsystem

图 10 需求变更管理子系统

在该子系统中,需求变更请求创建率(issue created rate)表示项目中每周需求变更请求创建的数量,单位为Dmnl,计算公式如(16)所示.

$$\left.\begin{array}{l} issue\ created\ rate = \\ \text{IF THEN ELSE}(\text{"}Base\text{-}line\ issue\ created\ rate\text{"} \geqslant 0 : \text{AND} : Number\ of\ users \geqslant 0, \text{"}Base\text{-}line\ issue\ created\text{"}, 0) \end{array}\right\} \tag{16}$$

处于“to do”状态的需求变更请求(number of to do issue)是需求变更等待实现的数量,单位为 issue,计算公式如(17)所示.

$$Number\ of\ to\ do\ issue = \text{INTEG}\left(\text{IF THEN ELSE}\left(\begin{pmatrix} accepted\ issue\ rate + Return\ to\ do\ rate - \\ issue\ change\ to\ sprint\ rate \end{pmatrix} \geqslant 0, \begin{pmatrix} accepted\ issue\ rate + Return\ to\ do\ rate - \\ issue\ change\ to\ sprint\ rate \end{pmatrix}, 0\right), 95\right) \tag{17}$$

已实现需求变更总数(resolved & closed issue)的单位为 issue,计算公式如(18)所示.

$$\text{“}resolved\ \&\ closed\ issue\text{”} = \text{INTEG}(\text{“}resolved\ \&\ closed\ rate\text{”}, 0) \tag{18}$$

无效需求变更请求百分比(percent of invalid issue)表示无效需求变更请求在需求变更总数所占百分比,单位为 Dmnl,计算公式如(19)所示.

$$\left.\begin{array}{l} percent\ of\ invalid\ issue = \\ \text{IF THEN ELSE}(TOTAL\ issue\ number \neq 0, Number\ of\ invalid\ issue / TOTAL\ issue\ number, 0) \end{array}\right\} \tag{19}$$

接受需求变更请求百分比(percent of accept issue)表示接受需求变更请求在需求变更总数所占百分比,单位为 Dmnl,计算公式如(20)所示.

$$\left.\begin{array}{l} percent\ of\ accept\ issue = \text{IF THEN ELSE}(TOTAL\ issue\ number \neq 0, \\ (Number\ of\ sprint\ issue + Number\ of\ to\ do\ issue + Number\ of\ waiting\ to\ rework\ issue + \\ Number\ of\ waiting\ to\ test\ issue + \text{"}resolved\ \&\ closed\ issue\text{"}) / TOTAL\ issue\ number, 0) \end{array}\right\} \tag{20}$$

1.4 开源软件需求变更管理系统动力学模型检测

建模工作完成后,对模型进行检测是为了保证模型结构和行为的有效性,确定模型与建模目的相符,发现模型的缺陷并提高模型的有用性.从建立模型真实性的角度来说,在系统动力学领域,仿真模型完全的有效性是不可能的,我们只能从仿真模型的目的出发,选择最为恰当的模型.因此在建模的过程中,我们从两方面来尽量保

证模型的有效性:一方面,通过和有关的专家进行讨论,确认模型是否符合实际情况,并对不符合的情况进行修改;另一方面,根据 Sterman[13]提出的 12 种用于系统动力学模型的检测方法,我们选择其中 11 种方法,通过实验对模型进行检测.检测目的、步骤和结果见表 1.

Table 1 Model test in system dynamics

表 1 系统动力学模型检测

检测方法[13]		检测目的	检测步骤/未检测原因	检测结果
1	边界充分检测	确保模型中包含重要的概念,评估模型边界是否与建模目的相符	从模型中移除重要的因果反馈环,看系统是否出现异常行为.本文删除一个反馈循环图后,经过仿真运行后发现异常,如处于“open”状态的 bug 数量为负导致模型行为出现异常,不符合现实系统 bug 数量不能为负的取值范围设定	模型通过边界充分检测
2	结构评估检测	确保模型符合质量守恒物理定律	1. 检查模型中的方程,根据结果对变量的取值范围进行控制. 2. 检测模型对于输入非法值的响应是否符合要求. 以需求变更管理子系统为例,对该系统中所有变量取极端值−1,即取值范围之外的值,检测结果显示,在输入非法值的情况下,与现实系统中变量不为负的取值范围要求一致	模型通过结构评估检测
3	量纲一致性检测	检查每个方程量纲是否一致	1. 使用Vensim自带的“Units Check”功能对模型进行量纲检测. 2. 人工逐个进行方程量纲检查. 在第一次量纲检测时发现模型中存在4个量纲未赋值或量纲不一致的错误,进行量纲修正后再次进行检测,所有模型符合量纲一致性	模型通过量纲一致性检测
4	参数评估检测	检查参数值是否与系统的相关描述一致且取值一致	完成该项检测需使用统计学方法和细化模型.但由于当前模型是软件需求变更管理过程是一个高度抽象,细化模型需要更多的项目细节和项目人员参与,故该部分检测将在下一步工作中完成	未检测
5	极端条件检测	确保变量在输入极端值的情况下,每个方程仍然有意义	1. 检查方程是否能处理极端输入值. 2. 检测极端输入值模型的响应.	模型通过极端条件检测
6	积分错误检测	时间单位选择应当合适,且具有做够的灵敏度	通过缩短模型的时间单位间隔来进行检测,本文中将时间间隔从以“周”为单位缩减为以“天”为单位来进行检测.以质量管理子系统中“bug response rate”为例,以“周”为单位,能更直观、精确反映 bug 在分支的生命周期中的响应数量的连续变化	以“周”为时间单位比以“天”为时间单位更合适
7	行为重现检测	检查模型是否重新系统中的重要的行为	使用 Matlab 2015b 中自带的逐点拟合工具对仿真结果与实际数据进行比较.以变量“bug create rate”为例,将仿真数据同现实数据进行逐点拟合分析,得到拟合优度 R^2=0.9716	R^2的值越接近 1,说明回归直线对观测值的拟合程度越好,检测结果表明拟合程度高
8	行为异常检测	通过检查删除或修改关系时模型是否出现异常行为来确定模型的重要关系	采用回路中断分析进行行为异常检测,通过屏蔽模型中各回路所带来的影响,看模型是否有异常行为.在模型中删除任意一个基于因果关系图的结构,都会导致行为异常,所以这些变量都是不可屏蔽或删除的	模型通过行为异常检测
9	家族成员检测	确定该模型是否能够在与系统相同类的其他实例中表现出不同行为	通过检测模型是否能生成在相同系统的其他例子中所观察到的现象来检测 本文除了对 Spring Framework 3.2.x 分支进行仿真建模外,还对 Hadoop 3.0.0 分支中的需求变更管理过程进行仿真建模,仿真结果表明,两个项目的多个变量均呈现不同的行为模式,说明模型具有能反映多种不同行为模式的能力,模型在所建模型的系统范围内具有一定的适用性	模型在不同案例中能表现出不同的行为,具有多样性
10	意外行为检测	检查模型中是否有意外行为发生	意外行为的发生是偶然且不可预期的,故本文只有当意外行为发生时,才进行检测	研究及分析期间,未发现模型的意外行为

Table 1 Model test in system dynamics (Contiuned)

表 1 系统动力学模型检测(续)

检测方法[13]		检测目的	检测步骤/未检测原因	检测结果
11	灵敏度分析检测	评估变化带来的影响,即当假设在合理的不确定范围内发生变动时,模型所得出的结论是否事关重大	以变量“review effectiveness”为例,分析采用代码评审的最差情况是经过代码评审后 bug 的报告量未减少,即评审有效性的变量取值为 0,最优的理想情况是经过代码评审后所有 bug 都被发现,即评审有效性的变量取值为 1,结合现实情况,评审有效性取值范围为[0,1),检测结果表明基线情况处于最优情况和最差情况之间,故评审有效性的基线情况通过单值灵敏度分析检测	模型通过灵敏度分析检测
12	系统改进检测	建模的过程帮助系统变得更好	从代码评审、激励措施的有效性和需求变更请求评估这 3 个方面对软件需求变更管理过程的改进进行仿真	仿真结果表明,采用软件过程改进策略,系统质量得到进一步提升

上述对模型的 11 种检测结果表明,模型均通过以上检测,达到建模目的.下面以 Spring Framework 项目为研究案例,使用版本分支 3.2.x 的数据进行需求变更管理过程的仿真,并对仿真结果进行分析改进.

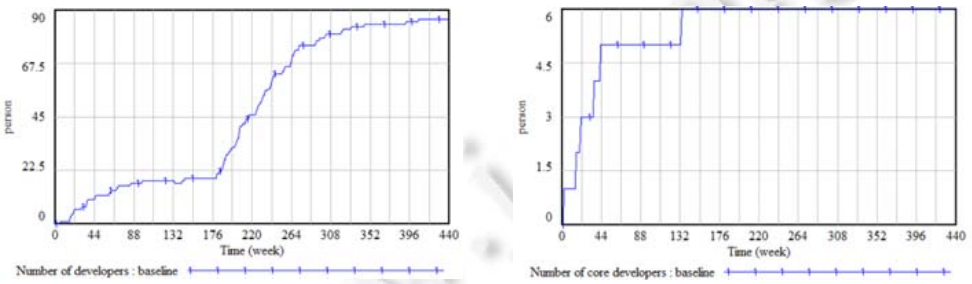
2 案例研究

Spring Framework 为任何部署平台上基于 java 的现代企业应用系统提供一个全面的编程和配置基础架构支持,由于其构建了一个企业应用程序的”管道”,使用这个框架的团队就可以专注于应用程序级别的业务逻辑,而不必在意特定的部署环境.另外,Spring Framework 项目在开源社区软件中是比较活跃且受众面较大的一个开源项目,相对而言经过长时间的运维,该软件较为成熟,且该项目采用了敏捷开发方法,本文的需求变更模型也具有敏捷开发的特征,因此选择 Spring Framework 项目为本文的研究案例.

在进行模型仿真前,对 Spring Framework 项目中的相关数据进行收集和清理:首先,收集 Spring Framework 的 issue tracking system(JIRA 系统)中所有 issue 数据,使用 git 工具以及 git clone 命令,通过 url 方式将 spring framework 的 commit 数据拉取到本地;之后,对于 issue 数据,在删除不需要的字段后,通过数据库导入工具将 issue 数据导入数据库;对于 commit 数据,使用 java 文本处理程序将数据导入数据库中;最后将所有数据导入到数据库,执行 Sql 脚本,查询导出仿真所需数据,组织为 csv 格式,并执行数据预处理程序,将数据转换成 Vensim 可以识别的数据格式并输入 Vensim,进行基线实验仿真分析.

2.1 基线实验仿真分析

将收集清理的数据在 Vensim 中进行仿真,得到如图 11~图 15 所示的基线仿真结果.此部分根据基线结果进行 Spring Framework 项目版本分支 3.2.x 需求变更管理过程分析,提出该项目需求变更管理过程的优缺点.



(a) 开发人员数量 (b) 核心开发人员数量

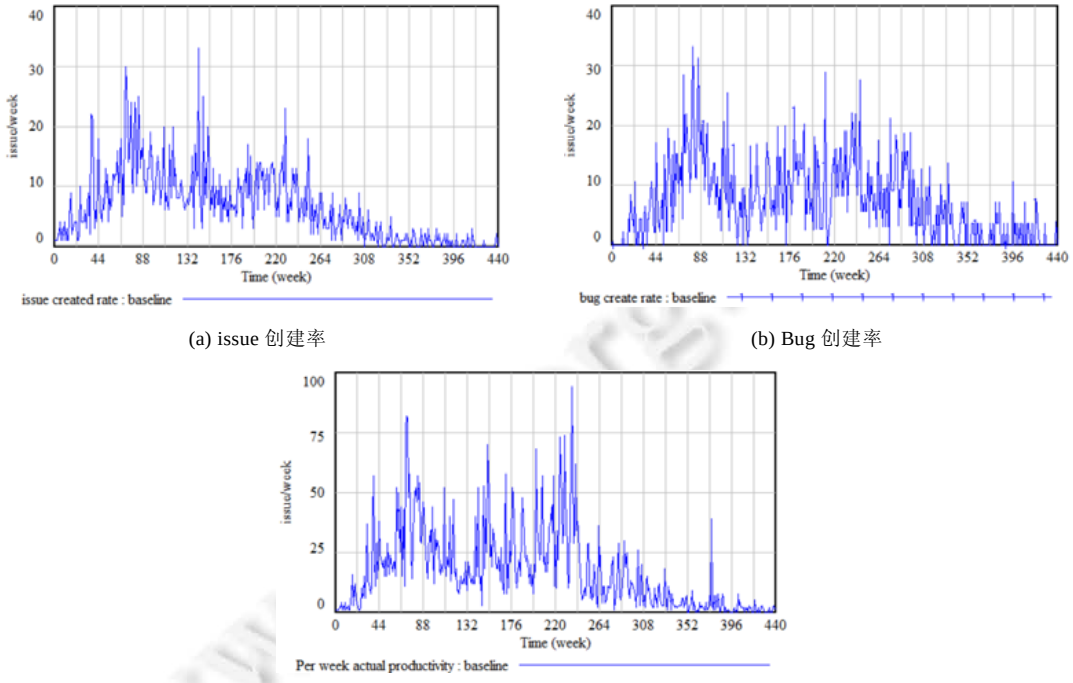
Fig.11 Baseline simulation results 1

图 11 基线仿真结果 1

图 11(a)和图 11(b)都是开源软件社区人力资源子系统的仿真结果,分别表示外围开发者以及核心开发团队成员的数量随时间的变化趋势.在分支 3.2.x 的生命周期中,两者都呈现出明显的增加趋势,反映出软件项目组

的逐渐发展壮大.

图 12(a)是需求变更管理子系统的仿真结果,表示需求变更每周创建的数量;图 12(b)是质量管理子系统的仿真结果,表示 bug 每周创建的数量;图 12(c)是进度计划与控制子系统的仿真结果,表示每周实际有效的需求变更生产力.



(a) issue 创建率 (b) Bug 创建率

(c) 每周实际生产力

Fig.12 Baseline simulation results 2

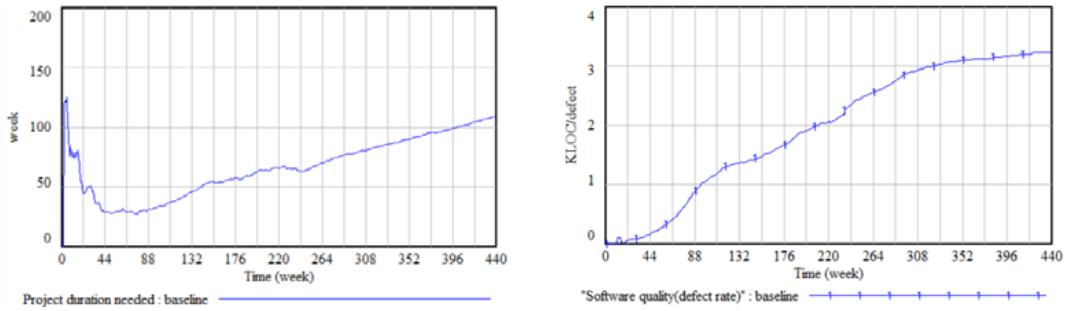
图 12 基线仿真结果 2

使用曲线数据拟合分析图 12(a)的需求变更请求创建率与图 12(c)中每周实际的生产力的相似性,根据拟合优度 R^2的值来判断拟合程度:R^2的值越接近 1,说明回归值对观测值的拟合程度越好.由分析结果得到:图 12(a)与图 12(c)的拟合优度 R^2>0.6.即需求变量请求的创建率与每周实际的生产力的相似性较高,需求变更请求的创建数量对开发人员的实际生产力有明显的影响.

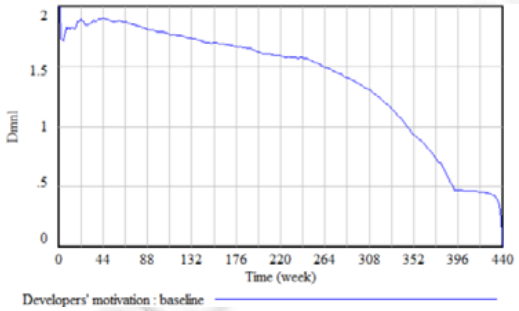
从需求变更对开发者生产力影响方面进行分析,根据图 12(a)和图 12(c)的仿真和拟合结果,在 Spring Framework 项目版本分支 3.2.x 需求变更管理过程中,项目组开发人员能够根据变更请求的情况作出积极及快速的响应.在该分支的前期,随着需求变更请求数量的快速增加,开发人员处理软件需求变更方面工作所花费的时间随之增加.在分支的中后期(项目的 244 周之后),由于 Spring Framework 4.x 版本的发布,以及需求变更请求的逐渐减少,开发人员在处理软件需求变更方面所花费的时间逐渐减少,直至项目分支被关闭.未得到解决的需求变更请求被迁移至其他版本分支中实现.因此在 244 周之后,项目开发人员的生产力也降低.

使用数据拟合分析图 12(b)和图 12(c),分析结果显示,其拟合优度为 R^2>0.3.即 bug 创建率与每周实际的生产力具有一定的相似性,bug 创建数量也影响着开发人员的实际生产力.由于实际情况中 Bug 数量的产生会有所滞后,因此开发人员的生产力变化也会有一定的时间差.

图 13(a)是进度计划与控制子系统的仿真结果,表示该分支处理需求变更仍需的持续时间;图 13(b)是质量管理子系统的仿真结果,用软件缺陷率的变化表示软件质量的变化;图 13(c)是开源软件社区人力资源子系统的仿真结果,表示开发人员的动机强度的变化.



(a) 项目持续时间 (b) 软件质量(缺陷率)



(c) 开发人员动机

Fig.13 Baseline simulation results 3

图 13 基线仿真结果 3

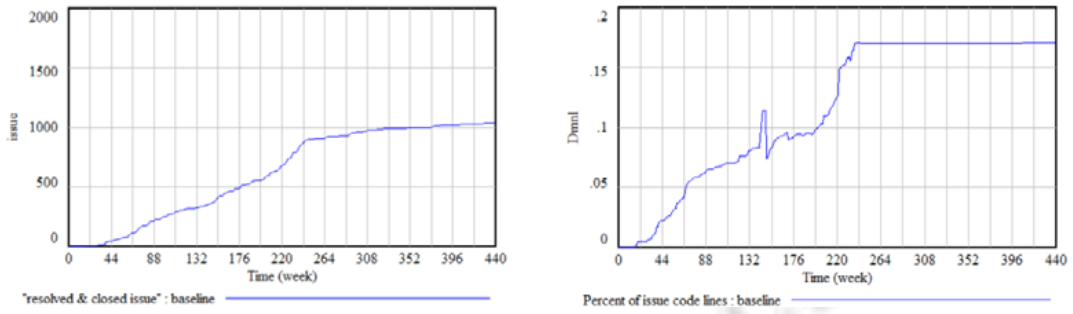
从需求变更对软件进度的影响方面,根据图 12(a)、图 12(b)与图 13(a)、图 13(b),在该需求变更管理过程中,随着时间的变化,处理软件需求变更所需的持续时间先降低后逐渐增加.处理软件需求变更所需的持续时间是由待完成的需求变更请求数量除以实际人员的平均生产力计算得到的,在分支前期,需求变更请求数量较少且开发人员的生产力快速上升,使得处理软件需求变更所需的持续时间降低;但在分支中期至后期,需求变更请求数量急剧增加,而开发人员平均生产力上升速度相对缓慢,使得处理软件需求变更所需的持续时间不断增加.这就造成分支的进度压力不断上升,从而使得开发人员动机强度逐渐下降(如图 13(c)所示),并近一步导致软件的代码质量缓慢下降(图 13(b)中软件缺陷率上升表示软件质量下降).需要说明的是,在 310 周时,版本 3.2.10 发布,bug 的报告率减少,故在 310 周后,软件的缺陷率变化较为平缓.

图 14(a)是需求变更管理子系统的仿真结果,表示已完成的需求变更请求 issue 数;图 14(b)是需求变更实现子系统的仿真结果,表示需求变更代码量占总代码行数的百分比.

从需求变更对软件规模的影响方面,根据图 14(a)和图 14(b),在该需求变更管理过程中,随着完成的需求变更数量的增加,软件变更的总代码行数不断上升,软件需求变更使得软件的规模扩张,增加了代码维护管理的难度;同时,需求变更所造成的代码变更百分比也随需求变更数量不断上升,代码的频繁变更和大范围变更也会造成软件的代码质量缓慢下降,如图 13(b)所示.另外需要说明的是,在 244 周后,版本 4.0 发布,此后项目组将大部分精力投入到了新版本 4.0.x 的开发中去.故在 244 周之后,项目的需求变更完成减少,代码的变化量也趋于平缓.

图 15(a)和图 15(b)都是需求变更管理子系统的仿真结果.图 15(a)表示接受的需求变更请求所占百分比,可以看出,该项目对用户提出的需求变更请求接受度均在 90%以上,因此该项目对于用户提出的需求变更有积极响应,对于吸引用户的增加有积极的作用.图 15(b)反映了无效、被拒绝的需求变更请求数量占总需求变更请求总数的百分比随时间的变化过程.在整个软件生命周期中,无效、被拒绝的需求变更请求在 10%的范围内.在项目分支前期,无效、被拒绝的需求变更请求百分比呈总体快速上升趋势;从项目分支中期开始,需求变更请求百

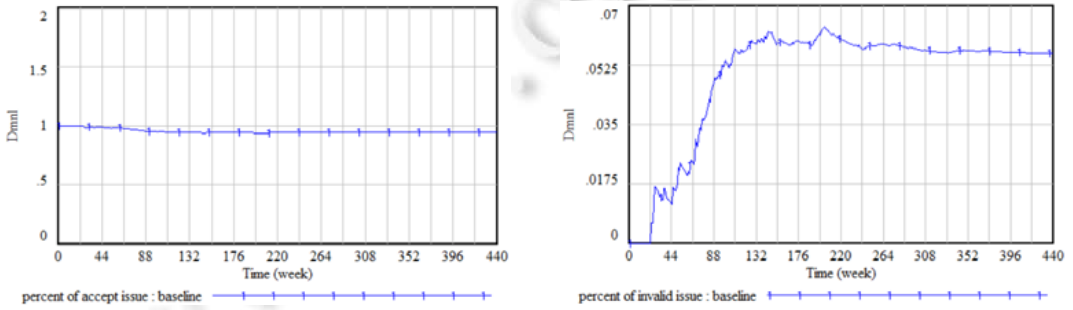
分比就在 6%上下浮动,且逐渐呈现出平稳的趋势.可见,项目组对无效、被拒绝的需求变更请求进行了比较有效的控制.



(a) 解决和关闭 issue 量 (b) Issue 代码行比例

Fig.14 Baseline simulation results 4

图 14 基线仿真结果 4



(a) 接收代码率 (b) 无效代码率

Fig.15 Baseline simulation results 5

图 15 基线仿真结果 5

综上,根据基线仿真结果的分析,Spring Framework 项目组在版本分支 3.2.x 的需求变更管理方面有许多优势,下面分析是否有可能进一步优化改进.

2.2 需求变更管理过程改进仿真

软件过程改进,就是采取策略来改进过程中被识别出来的薄弱环节,这些薄弱环节可能在改进前给软件质量、成本或是进度带来风险或缺陷[14].针对 Spring Framework 项目版本分支 3.2.x 需求变更管理过程中存在的缺陷率逐渐升高、开发人员动机强度逐渐下降等基线分析结果,下面通过软件代码评审环节以及增加开发人员动机强度管理进行过程改进仿真,同基线仿真结果作比较,分析改进的有效性.

2.2.1 增加代码评审环节

在 Spring Framework 软件项目的需求变更管理过程中,缺陷率在版本分支 3.2.x 的生命周期中不断上升,要有效地对软件的缺陷率进行控制,需要降低 bug 产生的数量以及进行合理的进度压力控制.从降低 bug 产生的数量方面来说,根据模型的因果关系图,可以增加代码评审环节,或是增强开发人员的动机强度,或者两者同时进行.

在基线数据中,代码的评审有效性(review effectiveness)设置为 0,即没有代码评审环节;现为降低 bug 的报告数量,模拟增加代码评审环节,将代码的评审有效性设置最小值为 0、最大值为 1、方差为 0.05、标准差为 0.1、种子数为 0 的随机正态分布函数,见表 2.

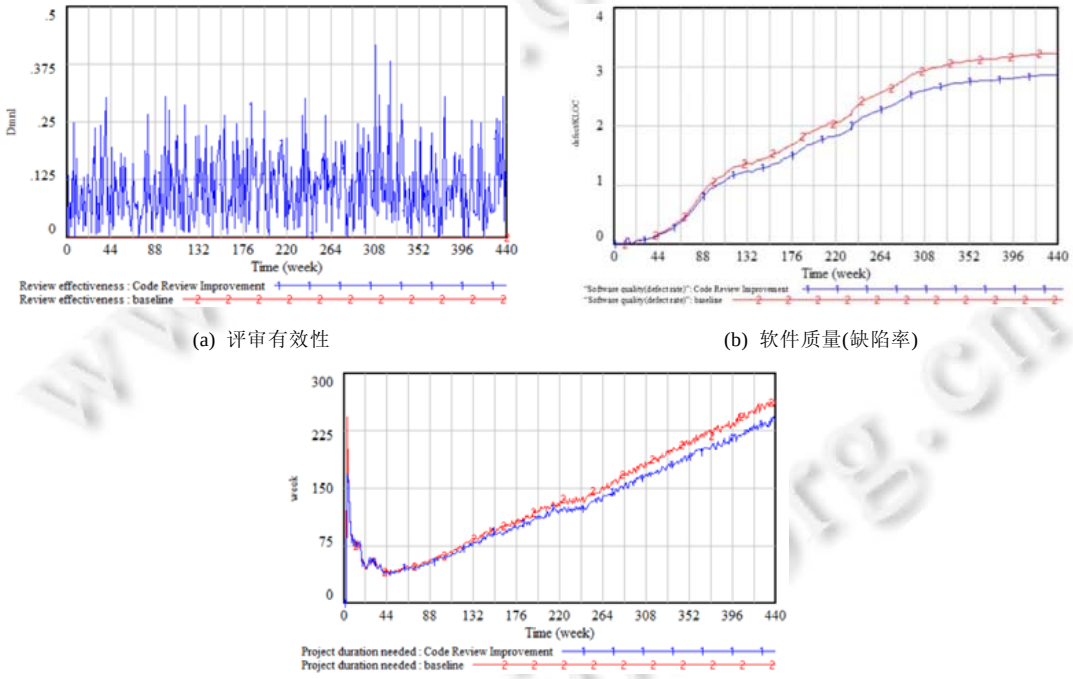
Table 2 Parameter setting for review effectiveness improvement

表 2 评审有效性改进参数设置

仿真名称	方程	方程释义
Baseline	0	表示基线结果没有采取代码评审环节.
Code review improvement	RANDOM NORMAL(0,1,0.05,0.1,0)	表示采取了代码评审环节,评审有效性呈随机正态分布.

将代码评审有效性参数更新后,进行改进仿真,得到如图 16 所示的改进结果.

- 图 16(a)是采取代码评审有效性的基线数据同改进数据的对比,图中红的线表示基线数据未采取代码评审环节时评审有效性随时间的变化趋势,即“Review effectiveness”变量函数为 0 时;蓝色的线表示采取代码评审有效性时评审有效性随时间的变化趋势,即“Review effectiveness”变量函数为 RANDOM NORMAL(0,1,0.05,0.1,0)时;
- 图 16(b)是软件缺陷率的基线数据同改进数据的对比,软件缺陷率越低,表示软件质量越高.图中红色的线表示基线数据未采取代码评审的软件质量随时间的变化情况;蓝色的线表示采取代码评审策略后软件缺陷率随时间的变化情况;
- 图 16(c)是软件项目所需的持续时间对比,图中红色的线表示基线数据中软件项目所需的持续时间;蓝色的线表示进行代码评审后软件项目所需的持续时间.



(a) 评审有效性 (b) 软件质量(缺陷率)

(c) 项目需要的持续时间

Fig.16 Simulation results for code review improvement

图 16 代码评审改进仿真结果

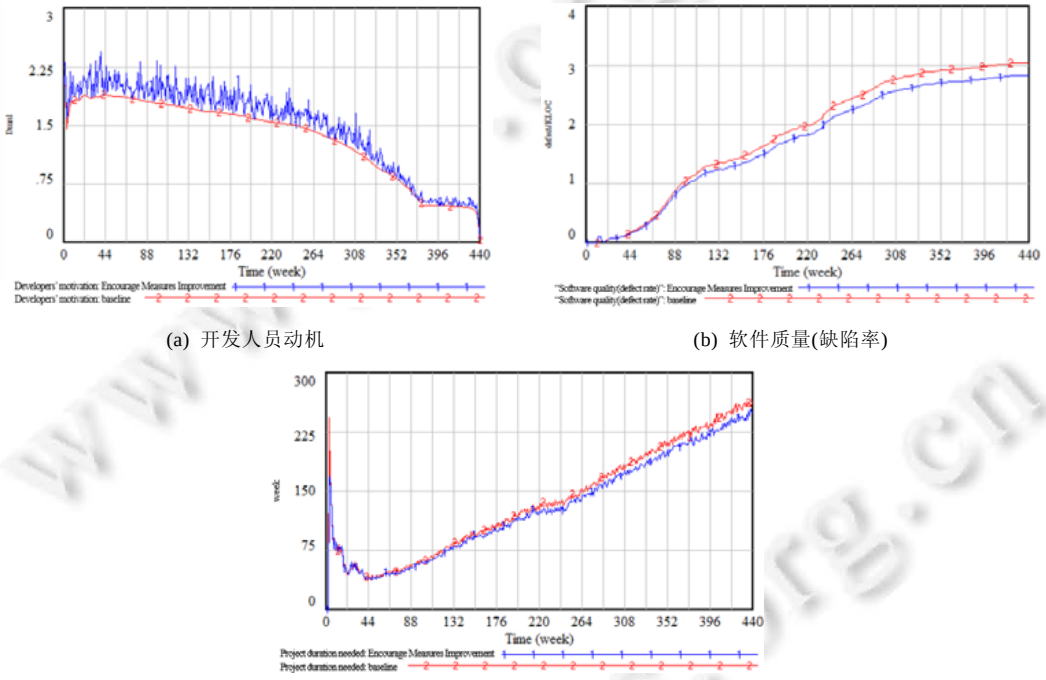
经过各子图基线数据同改进仿真数据对比,可以得出,增加代码评审环节可以降低基线数据的软件缺陷率,提高软件质量,并降低软件项目所需的持续时间.

2.2.2 激励开发人员的动机强度

大量文献研究表明,在开源软件项目中,开发人员的动机强度管理对软件的成败起着至关重要的作用[12].在案例的基线数据中,激励开发人员的动机强度的策略有效性(effectiveness of encourage measures)设置为 0,即没

有主动采取激励开发人员的有效策略.现对开发人员动机强度进行管理,模拟采取激励开发人员的有效策略,例如,可以根据贡献给予开发人员在团队中相应的地位和权利[15]、使用开发者的姓名来命名其所作的工作或者在社区中公开对其所作的工作进行表扬[16]以及进行匿名代码评审以减少开发人员受到的指责[12]等.将代码的动机强度的策略有效性方程更改为最小值为 0、最大值为 1、方差为 0.05、标准差为 0.1、种子数为 0 的随机正态分布函数,即与表 2 使用相同的随机函数.对上述变量进行调整后,再次进行改进仿真,得到如图 17 的人员动机强度改进结果.

- 图 17(a)是开发人员动机强度的基线数据同改进数据的对比,图中红色的线表示基线数据未采取激励策略时开发人员动机强度随时间的变化情况;蓝色的线表示采取激励策略后开发人员动机强度随时间的变化情况.
- 图 17(b)是软件缺陷率的基线数据同改进数据的对比,图中红色的线表示基线数据中的软件缺陷率随时间的变化情况;蓝色的线表示采取激励策略后软件的缺陷率随时间的变化情况.
- 图 17(c)是软件项目所需的持续时间的基线数据同改进数据的对比,图中红色的线表示基线数据中软件项目所需的持续时间;蓝色的线表示采用激励策略后软件项目所需的持续时间.



(a) 开发人员动机 (b) 软件质量(缺陷率)

(c) 项目持续时间

Fig.17 Simulation results for encourage measures improvement

图 17 激励策略改进仿真结果

经过各子图基线数据同改进仿真数据对比分析,可以得出,采取激励开发人员的策略可以降低基线数据的软件缺陷率,提高软件质量,并降低软件项目所需的持续时间.

2.2.3 增加需求变更请求筛选和分类

在软件需求变更管理过程中,无效、重复的变更会给软件的成本、时间造成损失.例如,无效、重复的变更可能会造成开发人员动机强度和软件质量的下降、处理需求变更所需时间延长等.为了降低无效、重复的变更给软件带来的损失,在 Spring Framework 项目版本分支 3.2.x 需求变更管理过程中,对需求变更请求进行筛选以及分类,可以有效识别出无效以及重复的变更请求(invalid & duplicate issue).为说明在软件需求变更管理过程

中对需求变更请求进行筛选和分类的必要性,本文对需求变更管理子系统再次进行了建模,得到如图 18 所示的模型.

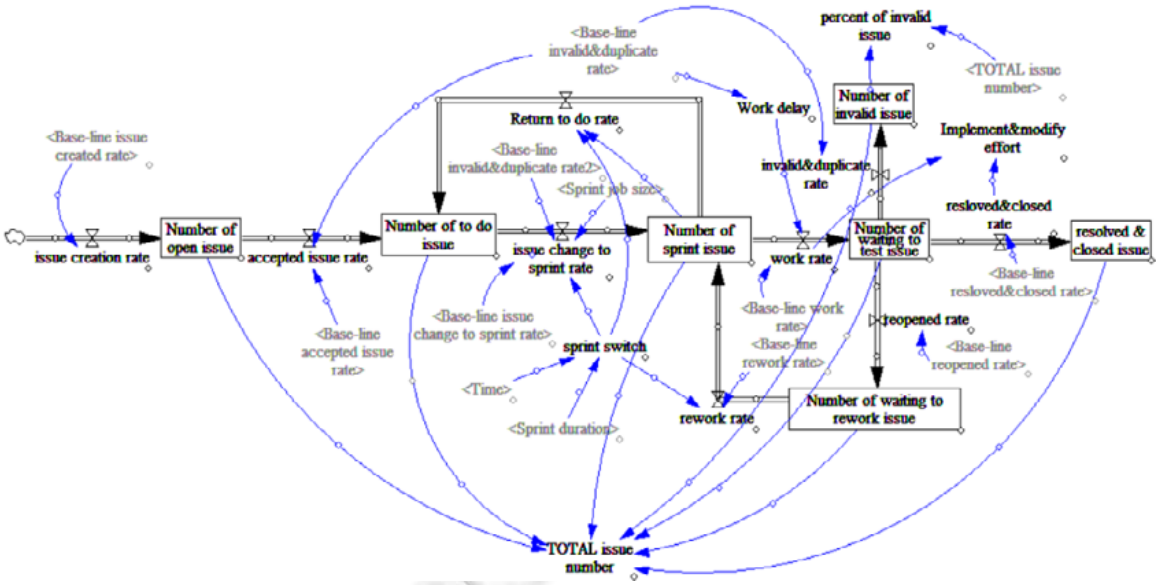
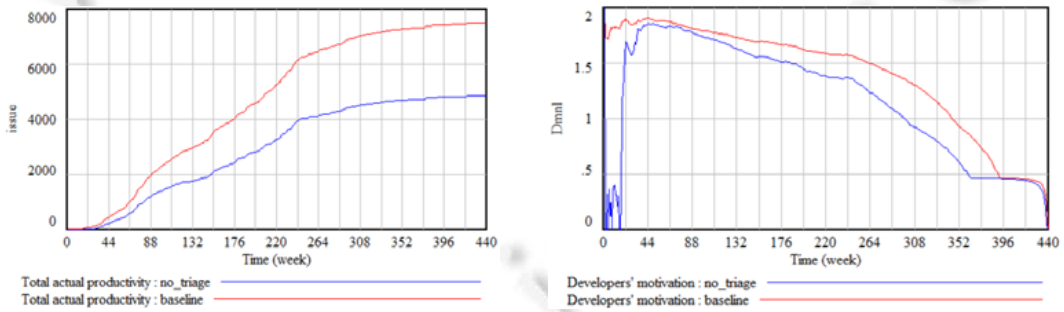


Fig.18 No_triage issue tracking subsystem

图 18 无需求变更请求筛选及分类的需求变更管理子系统

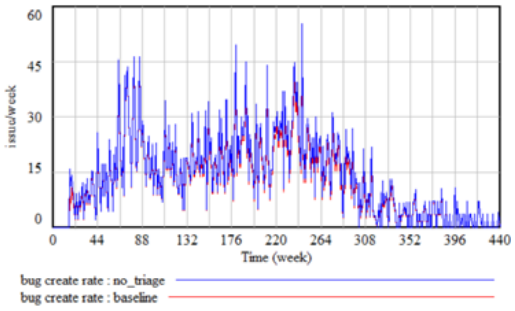
该模型与 2.3.5 节图 10 的需求变量管理子系统模型相比,主要不同在于没有进行需求变更请求筛选及分类.图 18 中的模型由于缺少需求变更请求筛选和分类环节,因此只要是需求变更请求就全部接受并直接进行变更管理,即无效、重复的需求变更请求都被直接执行变更,从而造成了在进行无效、重复的需求变更过程中投入的劳力、时间被浪费等后果.为了说明对需求变更请求进行筛选以及分类的必要性,对图 10 和图 18 中的两个需求变更管理过程的子系统模型进行仿真对比,得到如图 19 所示的结果.



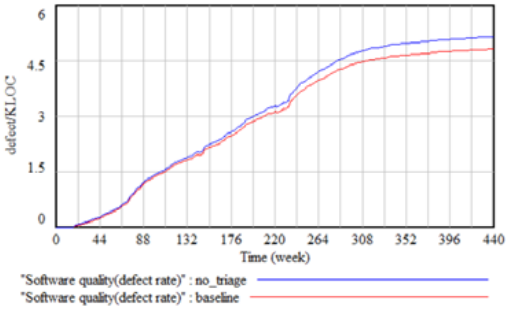
(a) 总体真实的生产力 (b) 开发者动机

Fig.19 Simulation results for no_triage

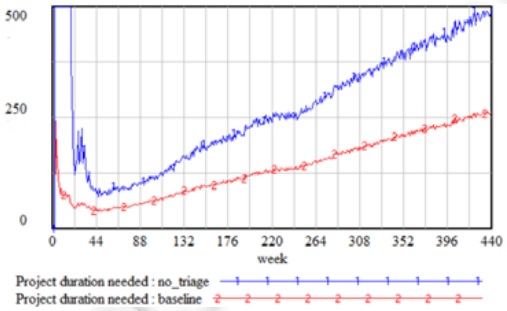
图 19 无需求变更请求筛选及分类的仿真结果



(c) bug 创建率



(d) 软件质量(缺陷率)



(e) 项目持续时间

Fig.19 Simulation results for no_triage (Continued)

图 19 无需求变更请求筛选及分类的仿真结果(续)

2.2.4 改进仿真结果对比分析

本节针对上述 3 种改进策略的仿真结果,通过分析 Spring Framework 采用 3 种策略所需的成本以及对应的改进有效性,提出改进建议.在提高相同软件质量,即减少相同缺陷率情况下,采取 3 种策略所需的人力、时间成本见表 3.

Table 3 Cost comparison for improvements

表 3 改进花费对比

改进策略	人力成本	时间成本
增加代码评审环节	高	高
激励开发人员的动机强度	一般	一般
增加需求变更请求筛选和分类	少	少

通常情况下,采取 3 种改进策略分别需要花费不同的人力成本以及时间成本.如若采用增加代码评审环节的改进策略,在需求变更数量较多时,因为需要对代码进行审阅,需要较多人力来进行代码评审,有可能使得增加代码评审环节需要花费比较多的人力成本和时间成本.而采用激励开发人员动机强度的改进策略时,通常开源软件项目均会采用如第 2.2.2 节描述的方式来提高其人员的动机强度,与其他两种改进策略相比,需要花费的人力和时间成本处于中间水平.当然,需要注意的是,采取这类改进策略提高开发人员的动机强度时,应充分考虑策略可能会无效甚至起反作用,使得开发人员的动机强度降低.而对于增加需求变更请求筛选和分类策略,由于当前大中型开源软件大多都使用 issue tracking system 进行需求变更管理,工具本身可以提供一定的变更请求筛选和分类能力,相对而言,此项策略需要花费的人力成本和时间可能较少,故该策略也被大多数的开源软件项目组所采用.

在 Vensim 中,采用上述 3 种改进策略对项目的进度影响进行仿真分析,仿真结果如图 20 所示.在图 20 中,右图是左图放大后的局部细节.左图中:灰色的线表示没有采取需求变更请求筛选和分类的项目所需持续时间;

蓝色的线表示采取需求变更请求筛选和分类的项目所需持续时间;红色的线表示的是采取代码评审项目所需的持续时间;绿色的线表示激励开发人员增加动机强度项目所需的持续时间.根据此仿真结果和表 3 的成本对比分析,当人力成本和时间成本相对紧缺,但项目剩余的时间相对宽裕时,应优先考虑节约成本.故在此情况下,应优先考虑增加需求变更请求筛选和分类的改进策略,其次再考虑采取提高开发人员动机强度的改进策略,最后考虑增加代码评审的改进策略.

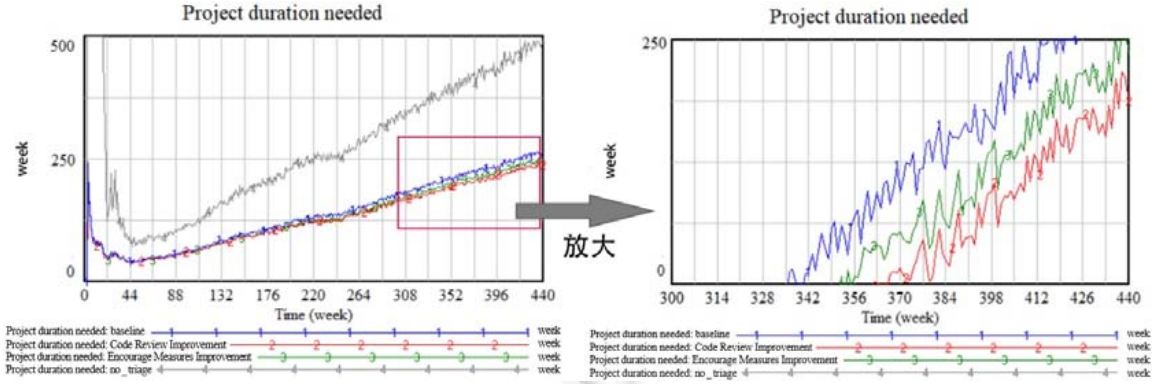


Fig.20 Comparison of project duration required for three improvements

图 20 采用 3 种改进所需项目持续时间对比

3 相关工作

目前的软件需求变更研究工作主要可分为软件需求变更产生原因分析、软件需求变更影响分析以及软件需求变更管理这 3 类.在软件需求变更产生原因分析方面,不同学者从不同的角度分析与总结了需求变更产生的原因.Bano 等人[17]基于已有文献分析,提出需求变更产生的原因可以面向业务、组织及项目视角按照根本原因和偶然原因进行分类.Nurmuliani 等人[18]使用卡片排序方法识别软件开发人员在实践中使用的软件需求变更请求类别.Kulk 等人[19]从定量的功能点计数中计算出需求波动率,并发现最大需求波动率取决于规模和持续时间.Mao 等人[20]构建需求变更因素分布评估框架,基于需求变更统计过程控制提出需求变更预测方法.在软件需求变更影响分析和变更管理方面,研究工作分为实证研究及仿真研究.实证研究人员通常通过收集实际软件开发项目的需求变更数据来进行变更影响分析.Nurmulian 等人[2]通过对多站点软件组织的两个软件项目的发布版本的实证研究,分析软件需求变更对软件开发工作的影响,并指出:在软件生命周期的末期进行软件需求变更,对软件的成本和进度来说是一个高风险的软件活动.而在仿真研究方面,研究者主要使用系统动力学仿真、离散事件仿真、Petri 网以及蒙特卡洛仿真等软件过程仿真方法进行相关研究[21].其中,使用系统动力学方法的文献在几种软件过程仿真方法中最多,文献[7,22−24]都使用系统动力学方法对软件需求变更进行研究.

系统动力学在软件工程领域方面的研究可追溯至 1964 年,Roberts 首先使用系统动力学建模了软件项目管理模型用来分析项目研究与开发[25].作为系统动力学在传统项目软件管理中的第一个经典的范例,Abdel-Hamid 与 Madnick 在 1991 年建立了 Abdel-Hamid 模型[26],该模型由人力资源管理、软件生产、计划和控制等部分组成,集成了软件开发过程的管理类型功能(例如计划、控制和人员配置)和软件生产类型活动(如设计、编码、评审和检测).此后,Abdel-Hamid 模型被应用到众多实际的软件项目管理和系统动力学在软件工程领域的研究中,至今仍具有巨大的应用和参考价值[27].除了 Abdel-Hamid 模型被大量用于软件项目管理以外,系统动力学还被广泛用于软件过程领域方面的研究.Madachy[28]使用系统动力学对软件过程的行为和结构进行分析,提出了软件过程动力学(software process dynamics).作为软件过程系统动力学仿真领域的经典著作,在该著作中,作者提供了大量工业界的案例来帮助人们学习软件过程动力学的使用.Kellner 等人[29]对软件过程仿真建模方法的文献综述中提到:根据研究目的,系统动力学软件过程仿真建模研究分为 6 类,分别是策略管理(strategic

management)、计划(planning)、控制与运行管理(control and operational management)、过程改进与技术适应(process improvement and technology adoption)、理解(understanding)和培训和学习(training and learning).与其他软件过程仿真方法相比,系统动力学所独有的因果反馈机制以及动态分析特点,使得系统动力学被大量运用于软件过程改进仿真建模研究.例如,Pfahl 等人[30]使用系统动力学方法建模了汽车自动化产业通用的战略软件过程改进仿真模型;Ruiz 等人[31]使用系统动力学方法仿真和建模了基于商用现货(commercial-off-the-shelf,简称COTS)的软件过程,以帮助理解这种软件开发过程的具体特点,并基于此模型设计和评价软件过程改进;Lin 等人[32]利用系统动力学的反馈原理,模拟软件生命周期开发活动和管理决策过程之间的动态交互,以帮助进行软件成本、进度和功能的度量;Ali 等人[33]面向两个真实的工业案例,提出一个辅助在精益管理方法中进行价值流分析的仿真框架,用于仿真在软件开发过程中可识别的过程改进,并基于仿真给出改进评估.

当然,系统动力学也常用于软件需求工程领域的研究,例如,Stallinger 等人[34]对软件开发早期阶段的需求活动的 EasyWinWin 过程进行建模和仿真;而在需求易变性(volatility)方面,Thakurta 等人[35]在软件需求波动情况下,分析需求波动人力变化对软件质量的影响;Ferreira 等人[7]使用系统动力学方法对需求波动进行建模,并对需求波动影响软件项目工作量、成本、时间等进行仿真分析;Cao 以及 Abdel Hamid 等人[24]对 Abdel-Hamid 模型进行调整,并加入了敏捷开发中所强调的结对编程、重构、用户参与、需求变更管理等软件活动,以使其能够适用于敏捷开发软件过程,并在案例中证明改进模型的有效性.Kellner 等人[21]以软件可靠性为研究目标进行了系统动力学的软件过程仿真建模,作为软件项目决策支持系统的一部分,用于协助项目经理以质量驱动的方式规划或定制软件开发过程.Antoniades 等人[36]使用动力学方法对开源软件的软件开发过程进行了一个通用的仿真模型建模,并使用 Apache 开源服务器中的开源项目数据进行仿真对比,说明其模型的有效性.Dutta 和 Roy[37]建立了技术和行为安全因素之间相互作用的系统动力学模型,并分析了它们对组织 IT 基础设施业务价值的影响.

此外,Lehman 提出软件演化的第一定律是持续的变更,因此,Lehman 的研究团队使用系统动力学方法进行了软件演化仿真方面的相关研究工作,在他们的研究工作中,Lehman 等人[38]、Wernick 等人[16]、Wernick 等人[39]、Kahen 等人[40]、Ali 等人[41]开发了多个持续改进的仿真模型,目的是通过仿真找出影响软件演化的主要因素以及辅助实现更好的软件演化管理.

总之,国外将系统动力学应用于软件领域的研究成果相对较为丰富,且为实际的软件项目节省客观的时间和成本,带来的巨大的经济效益.根据文献[5],系统动力学方法在软件领域的应用为将近 100 个软件项目节约了大约 80 亿美元的成本.国内应用系统动力学在软件工程领域的研究包括:何满辉等人[42]运用系统动力学方法建模软件项目进度管理并进行有效的预测,并分析人员配置对项目进度的影响;吴明辉[15]介绍基于系统动力学的连续型软件过程建模与仿真方法,并以 Brooks 法则作为实例描述该方法的基本要素和优势;翟丽等人[43]用系统动力学建模软件开发项目中各因素之间复杂的相互作用关系对项目绩效的影响,为项目管理者提供一个可靠的工具;杨勇等人[44]使用系统动力学方法对软件过程进行建模,并根据仿真结果计算偏离阈值;贾静[27]基于软件项目需求变更进行系统动力学建模,然后模拟不同条件下的需求变更对初始模型的影响,并在此基础上对初始模型进行改进,并进行需求变更控制的仿真和分析.

与上述相关研究工作相比,我们使用系统动力学方法不仅参考了敏捷过程,同时面向开源软件进行需求变更管理过程的仿真建模,通过使用真实开源软件项目数据作为模型的基线输入数据,为软件过程分析提供了一种有效的方法,并且较为完整地对模型进行了系统动力学检测,保证模型可用性.此外,我们还对软件需求变更管理过程进行了过程改进仿真,为控制项目成本和保证进度,提高软件质量,提供过程改进分析方法.相关研究方法和成果可应用于组织内软件项目,为项目组织团队保障软件生产的顺利实施提供决策建议.

4 总结与展望

本文首先使用系统动力学方法,参考敏捷过程的开源软件需求变更管理过程进行建模,并对模型进行检测以发现模型中的错误并改正;然后,以 Spring Framework 项目为研究案例,使用版本分支 3.2.x 的数据作为仿真模

型的基线数据,进行该分支的软件需求变更管理过程的仿真;接下来,根据仿真结果,对该分支的需求变更管理过程进行比较分析,提出了该需求变更管理过程中的不足;最后,模拟增加代码评审环节、激励开发人员的动机强度以及增加需求变更请求筛选和分类的需求变更管理过程改进策略,并根据仿真结果可以看出,3 种过程改进策略均可降低基线数据的软件缺陷率,提高软件质量并且使软件项目周期变短.在此基础上,基于投入成本与回报进行了 3 种改进策略的对比,并给出具体的改进实施建议.

总之,本文参考敏捷过程提出开源软件需求变更管理过程仿真模型.当前,国内外相关研究尚无相关研究文献发表,因此,通过分析总结开源软件需求变更管理过程的仿真研究结果,可以为组织内软件项目提供有价值的建议和经验.另外,当前使用真实软件项目数据来进行仿真研究的成果仍然较少,如果使用更多的真实数据进行案例分析,可以进一步优化仿真模型.因此,下一步的研究工作将收集更多的大中型开源软件需求变更管理数据,对模型进行模拟仿真分析和模拟改进分析,增加模型的适用性.

References:

[1] Curtis B, Krasner H, Iscoe N. A field study of the software design process for large system. Communication of the ACM, 1988, 31(11):1268−1287.

[2] Nurmuliani N, Zowghi D, Williams SP. Requirements volatility and its impact on change effort: Evidence-based research in software development projects. In: Proc of the 11th Australian Workshop on Requirements Engineering. Adelaide 2006. 1−10.

[3] Williams BJ, Carver J, Vaughn RB. Change risk assessment: Understanding risks involved in changing software requirements. In: Proc. of the Int'l Conf. on Software Engineering Research and Practice & Conf. on Programming Languages and Compilers. Trier Tele: DBLP, 2008. 966−971.

[4] Zowghi D, Nurmuliani N. A study of the impact of requirements volatility on software project performance. In: Proc. of the Software Engineering Conf. Piscataway: IEEE, 2002. 3−11.

[5] Godlewski E, Cooper K. System dynamics transforms fluor project and change management. Interfaces, 2012,42(1):17−32.

[6] Zhang H, Raffo, D, Birkhöltzer T, Houston D, Madachy R, Münch J, Sutton Jr SM. Software process simulation—At a crossroads? Journal of Software: Evolution and Process, 2014,26(10):923−928.

[7] Ferreira S, Collofello J, Shunk D, Mackulak G. Understading the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. The Journal of Systems and Software, 2009,82(10):1568−1577.

[8] Crowston K, Howison J, Annabi H. Information systems success in free and open source software development: Theory and measures. Software Process Improvement and Practice, 2006,11(2):123−148.

[9] Fuggetta A. Open source software-an evaluation. The Journal of Systems and Software, 2003,66(1):77−90.

[10] Yang B, Yu Q, Zhang W, Wu J, Liu C. Influence factors correlation analysis in GitHub open source software development process. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1330−1342 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5222.htm [doi: 10.13328/j.cnki.jos.005222]

[11] Heck P, Zaidman A. An analysis of requirements evolution in open source projects: Recommendations for issue trackings. In: Proc. of the Int'l Workshop on Principles of Software Evolution. 2013. 43−52.

[12] Aberdour M. Achieving quality in open source software. IEEE Software, 2007,24(1):58−64.

[13] Sterman J. Business Dynamics: Systems Thinking and Modeling for A Complex World. McGraw-Hill Companies, Inc., 2000.

[14] Dorling A. SPICE: Software process improvement and capability determination. Software Quality Journal, 1993,2(4):209−224.

[15] Wu MH. Modeling and Simulation of software process based on system dynamics. Computer Era, 2007,(9):1−4 (in Chinese with English abstract).

[16] Wernick P, Lehman MM. Software process white box modeling for FEAST/1. Journal of Systems and Software, 1999,46(2-3): 193−201.

[17] Bano M, Imtiaz S, Ikram N, *et al*. Causes of requirement change—A systematic literature review. In: Proc. of the Int'l Conf. on Evaluation & Assessment in Software Engineering. London: IET, 2012. 22−31.

[18] Nurmuliani N, Zowghi D, Williams S P. Using card sorting technique to classify requirements change. In: Proc. of the IEEE Int'l Requirements Engineering Conf. IEEE Computer Society, 2004. 240−248.

[19] Kulk GP, Verhoef C. Quantifying requirements volatility effects. Science of Computer Programming, 2008,72(3):136−175.

[20] Mao C, Lu Y, Wang X. A study on the distribution and cost prediction of requirements changes in the software life-cycle. In: Proc. of the Unifying the Software Process Spectrum. Berlin, Heidelberg: Springer-Verlag, 2006. 136−150.

[21] Kellner MI, Madachy RJ, Raffo DM. Software process simulation modeling: Why? What? How? Journal of Systems & Software, 1999,46(2-3):91−105.

[22] Stallinger F, Grünbacher P. System dynamics modelling and simulation of collaborative requirements engineering. Journal of Systems & Software, 2001,59(3):311−321.

[23] Thakurta R, Suresh P. Impact of HRM policies on quality assurance under requirement volatility. Int'l Journal of Quality & Reliability Management, 2012,29(2):194−216.

[24] Cao L, Ramesh B, Abdel-Hamid T. Modeling dynamics in agile software development. ACM Trans. on Management Information Systems (TMIS), 2010,12:Article No.5.

[25] Baer RE. The Dynamics of Research and Development. New York: Harper & Row, 1964.

[26] Abdel-Hamid T, Madnick SE. Software Project Dynamics: An Integrated Approach. Prentice-Hall, Inc., 1991.

[27] Jia J. The research on the impact of requirement change in software project based on system dynamic [MS. Thesis]. Tianjin: Nan Kai University, 2014 (in Chinese with English abstract).

[28] Madachy RJ. Software Process Dynamics. John Wiley & Sons, 2007.

[29] Kellner MI, Madachy RJ, Raffo DM. Software process simulation modeling: Why? What? How? Journal of Systems and Software, 1999,46(2-3):91−105.

[30] Pfahl D, Stupperich M, Krivobokova T. PL-SIM: A generic simulation model for studying strategic SPI in the automotive industry. In: Proc. of the Int'l Workshop on Software Process Simulation and Modeling. 2004. 149−158.

[31] Ruiz M, Ramos I, Toro M. Using dynamic modeling and simulation to improve the COTS software process. In: Proc. of the Product Focused Software Process Improvement. Berlin, Heidelberg: Springer-Verlag, 2004.

[32] Lin CY, Abdel-Hamid T, Sherif JS. Software-Engineering process simulation model (SEPS). Journal of Systems & Software, 1997, 38(3):263−277.

[33] Ali NB, Petersen K, de Franca BBN. Evaluation of simulation assisted value stream mapping for software product development: Two industrial cases. Information Software Technology, 2015,68(12):45−61.

[34] Stallinger F, Grünbacher P. System dynamics modelling and simulation of collaborative requirements engineering. Journal of Systems & Software, 2001,59(3):311−321.

[35] Thakurta R, Suresh P. Impact of HRM policies on quality assurance under requirement volatility. Int'l Journal of Quality & Reliability Management, 2012,29(2):194−216.

[36] Antoniades IP, Stamelos I, Angelis L, *et al*. A novel simulation model for the development process of open source software projects. Software Process Improvement & Practice, 2002,7(3-4):173−188.

[37] Dutta A, Roy R. Dynamics of organizational information security. System Dynamics Review, 2008,24(3):349−375.

[38] Lehman MM, Ramil JF. The impact of feedback in the global software process. Journal of System and Software, 1999,46(2-3): 123−134.

[39] Wernick P, Hall T, Nehaniv CL. Software evolutionary dynamics modeled as the activity of an actor-network. IET Software, 2008, 2(4):321−336.

[40] Kahen G, Lehman MM, Ramil JF, Wernick P. System dynamics modeling of software evolution processes for policy investigation: Approach and example. Journal of Systems and Software, 2001,59(3):271−281.

[41] Ali SM, Doolan M, Wernick P, Wakelam E. Developing an agent-based simulation model of software evolution. Information and Software Technology, 2018,96:126−140.

[42] He MH, Yang JP. Software project schedule management based on system dynamics. Science Technology and Industry, 2007,7(5): 11−13 (in Chinese with English abstract).

[43] Zhai L, Song XM, Xin YF. Software project management: Evaluation the problem solutions. Soft Science, 2008,22(1):59−62 (in Chinese with English abstract).

[44] Yang Y, Zhou BS. Software process deviation control based on system dynamics. Computer Engineering and Design, 2011,32(5): 1684−1686 (in Chinese with English abstract).

附中文参考文献:

[10] 杨波,于茜,张伟,吴际,刘超.GitHub 开源软件开发过程中影响因素的相关性分析.软件学报,2017,28(6):1330−1342. http://www.jos.org.cn/1000-9825/5222.htm [doi: 10.13328/j.cnki.jos.005222]

[15] 吴明晖.基于系统动力学方法的软件过程建模与仿真.计算机时代,2007,(9):1−4.

[27] 贾静.基于系统动力学的软件项目需求变更影响研究[硕士学位论文].天津:南开大学,2014.

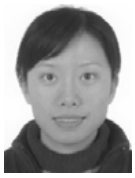
[42] 何满辉,杨皎平.基于系统动力学的软件项目进度管理.科技和产业,2007,7(5):11−13.

[43] 翟丽,宋学明,辛燕飞.系统动力学在软件项目管理中的应用:对解决问题各备选方案的评价.软科学,2008,22(1):59−62.

[44] 杨勇,周伯生.基于系统动力学的软件过程偏离控制.计算机工程与设计,2011,32(5):1684−1686.

康燕妮(1992－),女,硕士,主要研究领域为软件过程.

李彤(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件过程,软件工程.



张璇(1978－),女,博士,副教授,CCF 专业会员,主要研究领域为需求工程,软件过程,可信软件.

唐子淇(1995－),女,硕士,主要研究领域为软件过程.

王旭(1976－),男,博士,讲师,主要研究领域为金融安全,区域经济,计量经济学.

牛家梅(1994－),女,硕士,主要研究领域为软件过程.