需求变更技术债务研究

张云洁[1] 张 璇[1,2] 丁 浩[1] 王 旭[3]
(云南大学软件学院 昆明 650504)[1] (云南省软件工程重点实验室 昆明 650504)[2]
(云南大学经济学院 昆明 650504)[3]

摘 要 在软件生命周期中,需求不断发生变更,而需求决策往往取决于开发人员的偏好和权衡,缺乏一种系统的、明确的管理方法。针对软件生命周期中由不断出现的需求变更引起的技术债务,提出一种需求变更技术债务定义。通过对需求变更技术债务的定义、检测、量化和排序,为需求变更的实现顺序以及实现方式提供技术支持。最后通过实验验证了需求变更技术债务的概念和技术的可行性。

关键词 技术债务,需求变更,需求变更技术债务
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.09.013

Study on Technical Debt Caused by Requirement Change

ZHANG Yun-jie[1] ZHANG Xuan[1,2] DING Hao[1] WANG Xu[3]
(School of Software, Yunnan University, Kunming 650504, China)[1] (Yunnan Software Engineering Key Laboratory, Kunming 650504, China)[2]
(School of Economics, Yunnan University, Kunming 650504, China)[3]

Abstract In software life cycle, the requirement changes constantly, and the requirement decision often depends on developer's preference and balance, and lacks a systematic and definite management method. A definition of technical debt caused by requirement change was proposed for the technical debt caused by the constant requirement change in software life cycle. Through the definition, detection, quantification and sorting of requirement change technical debt, technical support was provided for the realization sequence and realization way of requirement change. Finally, experiments were conducted to verify the feasibility of concept and technology for technical debt caused by requirement change.

Keywords Technical debt, Requirement change, Requirement change technical debt

1 引言

在软件开发过程中,开发者为了快速达到一个短期的目标,可能会选择暂时忽略需求变更产生的影响,在开发中走"捷径"[1]。技术债务(Technical Debt)就是用来比喻因选择"捷径"所导致的长期维护代价而逐渐增加的后果,其最初仅涉及软件实现(即代码级),但目前已逐步扩展到软件体系结构、系统的详细设计,甚至文档、需求和测试等方面[2]。目前,软件系统变得越来越复杂,随着业务需求的演变,软件系统的需求不断发生变化,新的需求不断出现。研究表明,85%~90%的软件开发预算都用于运行和维护[3]。为了减少变更成本,尽早地管理需求变更非常重要。因为估算失误而推迟某些需求变更的实现从而影响其他需求变更的实现,以及因为工期限制被迫快速实现某些需求变更而引入新的技术债务,都会对软件长期的健康发展造成不可预知的影响,称之为需求变更技术债务。

本文着眼于研究需求变更所导致的技术债务,定义需求变更技术债务,根据债务产生方式的不同提出不同分类,并给出需求变更技术债务量化方法,以协助项目组管理技术债务并且做出相应决策。本文所提方法已被运用到开源项目HADOOP中,以实现项目中需求变更技术债务的量化和排序,验证了需求变更技术债务概念的可用性及技术的可行性。

本文第2节总结目前国内外对需求相关技术债务研究的相关工作;第3节介绍需求变更技术债务的定义及量化方法;第4节将本文提出的方法在Hadoop项目上进行实验验证。

2 相关研究

近年来,技术债务受到了学术界和产业界的广泛关注。技术债务最初就是Brown等所说的"长期软件质量与短期利益的权衡"[4]。该概念是由沃德·坎宁安(Ward Cun-

到稿日期:2017-10-25 返修日期:2018-01-09 本文受国家自然科学基金项目(61502413,61262025,61379032,61662085),云南省科技计划项目(2016FB106),云南省教育厅科学研究基金重点项目(2015Z020),云南省软件工程重点实验室开放基金项目(2015SE202),云南省创新团队"数据驱动的软件工程创新团队"项目,云南大学高水平创新团队"软件工程创新团队"专项项目资助。

张云洁(1994-),女,硕士生,主要研究方向为需求工程、软件工程经济学;张 璇(1978-),女,博士,副教授,主要研究方向为需求工程、软件过程、可信软件,E-mail:zhxuan@ynu.edu.cn(通信作者);丁 浩(1992-),男,硕士,主要研究方向为需求工程;王 旭(1976-),男,博士,副教授,主要研究方向为金融安全、计量经济学。

ningham)(wiki 的创始人)于 1992 年提出的,用来描述开发人员在短期收益和长期的软件健壮之间的权衡[5]。SonarQube 是一个用来管理代码质量的开放的平台,它可以快速发现项目和/或组件的技术债务,以建立行动计划。截止到 2017 年,已有 1600 篇文献涉及到技术债务,研究内容从代码方面逐渐扩展到软件工程所涉及的多种类型数据和整个软件开发周期,例如需求债务、架构债务、设计债务、测试债务等[1]。软件需求固有的不确定性(如客户或市场需求的不确定性)使得需求债务管理更具挑战[6]。Neil 等将需求债务定义为需求问题最优解和实际解决方案之间的距离,而将债务的利息定义为这一距离的增长速度,他还利用一种需求建模工具 rekombine 来管理不断变化的需求问题[7]。Zahra 等将实物期权的思想应用于软件开发项目中的需求债务管理,并采用二项式模型与动态规划相结合的方法论证了实物期权在需求债务评估中可行[8]。随着软件系统的不断变化,新的需求不断出现。当引入新的需求变更时,若只分析单个需求变更,则可能会忽略变更的实际影响,使实施成本比预期高出很多倍。Goknil 等提出了一种需求元模型,使用需求关系和需求变更类型的形式化语义来改进需求的变更影响分析[9]。Jameleddine 等在 Use Case Maps(UCM)上使用约束和依赖分析来确定需求变化对整个系统的影响,并且使用一个简单的电话系统进行案例分析,验证了该方法的适用性[10]。目前,学术界和产业界还没有需求变更技术债务的相关研究成果。

3 需求变更技术债务

3.1 需求变更技术债务的定义

在软件生命周期的各个阶段,每一个需求变更的提出都有一个潜在的需求变更技术债务,需求变更之间的依赖关系使得需求变更变得更加复杂,没有依赖关系也就不存在“利息”。为了直观地展示需求变更之间的关联关系,使用一个有向图来描述。其中,各个节点代表了不断提出的需求变更,而边代表了需求变更之间的关联关系。需求变更关联关系图的定义如下。

定义 1(需求变更关联关系图) 需求变更关联关系图是一个三元组,$G=\langle V,E,R\rangle$。其中 V 是节点的集合,v_i($v_i\in V$,$i=1,2,\cdots,n$)表示一个变更需求;E 是边的集合,e_i($e_i\in E$,$i=1,2,\cdots,n$)表示一个有序元素对(v_i,v_j)($v_i,v_j\in V$,$i=1,2,\cdots,n$;$j=1,2,\cdots,n$);R 是一个关联函数,它使得 E 中的每一个元素(称为有向边或弧)对应 V 中的一个有序元素(称为顶点或点)对,r_i($r_i\in E$,$i=1,2,\cdots,n$)表示一个关联关系。

在软件需求变更的管理工作中,大多数决策由开发人员的偏好和直觉决定[11],而且技术人员与用户之间的沟通也会存在理解偏差[12],由于软件开发人员无法准确地估算出需求变更给软件带来的影响和收益,导致某些需求变更没有实现或快速实现,从而为软件开发带来不可预计的后果。将这种为了实现短期利益而忽略长期发展的需求变更所带来的技术债务定义为需求变更技术债务,其由两部分组成:本金和利息。需求变更技术债务的定义如下。

定义 2(需求变更技术债务) 需求变更技术债务是指需求变更快速实现或没有实现所带来的花费(使用需求变更的解决时间减去提出时间来表示花费),分为本金和利息。其中,本金是指需求变更本身的花费,利息是指与该需求变更相关联的其他需求变更的花费。

将需求变更分为 3 种类型:一种是不存在变更关联关系的需求变更;另外两种是存在变更关联关系的需求变更,其中一类是如果不实现会影响其他需求变更实现,另外一类是快速实现造成的后期维护所花费的需求变更。需求变更的基本类型的定义如下。

定义 3(需求变更的基本类型) 需求变更的基本类型有 3 种:一般类需求变更、影响类需求变更、产生类需求变更。其中一般类需求变更是指不存在变更关联关系的需求变更;影响类需求变更是指如果不实现会影响其他需求变更实现的需求变更,将影响类需求变更的关联关系定义为 $affect$;产生类需求变更是指由快速实现而造成的后期维护花费的需求变更,将产生类需求变更的关联关系定义为 $produce$。

随着应用环境的不断变化,软件的更新速度越来越快,需求的变更不可避免。自顶向下的开发模式(在需求设计阶段直接设计出完美的软件蓝图)已经成为过去。而仅仅对需求变更技术债务进行定义,仍然不能确切地对需求变更技术债务有一个直观的认识,还需要对需求变更技术债务进行量化,只有更加具体地表现需求变更技术债务,才能更有效地管理需求变更。

3.2 需求变更技术债务的量化

需求变更技术债务的量化并非通过一个简单的公式即可实现,需要遵循一定的步骤,需求变更技术债务量化过程的具体步骤如图 1 所示。

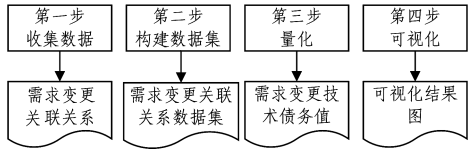

图 1 需求变更技术债务的量化步骤

Fig. 1 Quantification steps of technical debt caused by requirement change

1)从项目中收集历史需求变更数据,包括需求变更的名称和需求变更之间的关联关系。

根据 3.1 节中需求变更的基本类型的定义,将需求变更分为 3 种基本类型,图 2 给出了这 3 种不同需求变更的花费随时间的变化情况。其中横坐标表示时间,t 代表 issue 的不同阶段;纵坐标表示需求变更的花费(本文利用时间花费表示需求变更的花费),花费的增长意味着债务的积累。

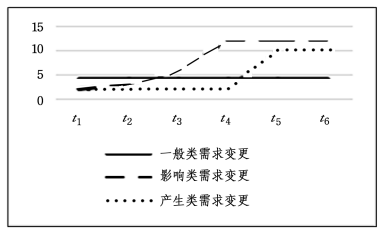

图 2 各类型需求变更的花费

Fig. 2 Change costs of different types of requirements

图 2 中，一般类需求变更表示与其他需求不具有关联关系的需求变更，只包含自身的花费，在图中表现为花费固定不变。影响类需求变更表示影响其他需求变更实现的需求变更，随着受影响的需求变更的不断增多，花费不断增长，债务值不断上升。如图 2 所示，到达 t_4 阶段后，受影响的需求变更数量达到最大值，债务值不再积累。影响类需求变更在需求变更关联关系图中表示为有序元素对(v_i,v_j)的关联关系 $r_k=affect$(其中 $i=1,2,\cdots,n;j=1,2,\cdots,n;k=1,2,\cdots,n$)。产生类需求变更表示因被迫快速实现而导致后期维护费用增长的需求变更。其虽然实现了需求变更，但在后期引发了一系列问题，导致维护费用不断增长。如图 2 所示，到达 t_4 阶段之后，由于一系列问题的出现，维护花费增加。到达 t_6 阶段后，由于引发的问题已经全部出现，维护花费不再增加，债务值保持不变。产生类需求变更在需求变更关联关系图中表示为有序元素对(v_i,v_j)的关联关系 $r_k=produce(i=1,2,\cdots,n;j=1,2,\cdots,n;k=1,2,\cdots,n)$。

2)从收集的数据中筛选出研究需要的需求变更和需求变更间的关联关系，构成关联关系数据集。通过关联关系数据集构建出需求变更关联关系图。

以 HADOOP 项目中的变更请求 HADOOP-4487 为例，HADOOP-4487 是第一个请求实现 HADOOP 的安全机制的变更请求报告，时间为 2008 年 10 月 22 日。HADOOP-4487 并没有被快速解决，随后，安全方面的变更请求不断被提出。图 3 为 HADOOP-4487 需求变更关联关系图。

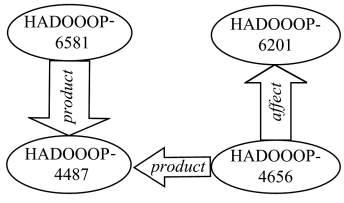

图 3　HADOOP-4487 需求变更的关系图

Fig. 3　Relationship of HADOOP-4487 requirement change

由图 3 可知，$V=$｛HADOOP-4487，HADOOP-4656，HADOOP-6201，HADOOP-6581｝；$r_1=produce$，表示 HADOOP-4487 和 HADOOP-6581 之间的的关联关系为 *produce*，依此类推。

3)计算出项目中每个需求变更技术债务的本金和利息，将本金和利息相加，就可以得到每个需求变更技术的债务值。

本文采用时间花费来度量需求技术债务，变更请求的时间花费为解决时间（Resolved Date）减去提出时间（Created Date）。需求变更时间花费的公式为：

$$C=T_{RE}-T_{CR}$$

其中，C 表示这个变更请求的时间花费，T_{RE} 表示变更请求的解决时间点，T_{CR} 表示变更请求的提出时间点。

将需求变更技术债务量化为两部分：本金和利息。需求变更技术债务的公式如下：

$$RD=C_R+\sum_{i=1}^{n}\sum_{j}^{m}l_i r_x C_{ij}$$

其中，RD 表示需求变更技术债务总值。C 表示需求变更所需花费，本文采用时间花费度量；C_R 表示需求变更技术债务本身的花费，即债务的本金。C_{ij} 表示需求变更所影响的第 i 层$(1\leqslant i\leqslant m)$、第 j 个方面的花费，即需求变更技术债务的利息。系数 l_1 表示需求变更之间直接关联时的强度系数，l_2 表示间接关联时第一层间接关联的强度系数，以此类推。系数 r_x $(1\leqslant x)$表示不同的依赖关系类型。这两个系数分别从关系层级和关系类型两方面描述了需求变更之间联系的强度。

4)对需求变更技术债务的量化结果进行可视化，从而可以直观地看出项目中需求变更技术债务的情况，利于理解和分析。

4　案例分析

HADOOP 是一个能够对大量数据进行分布式处理的软件框架。HADOOP 自 2006 年 1 月 28 日诞生至今已有 12 年，它改变了企业对数据的存储、处理和分析的过程，加速了大数据的发展，形成了自己的技术生态圈，并得到非常广泛的应用，具有一定的代表性和研究价值[13]。目前，HADOOP 通过 JIRA 系统来跟踪管理变更请求（Change Requirement），由用户或者开发人员提交到变更请求追踪系统上的变更请求从本质上反映了需求变更[14]。截至 2017 年，HADOOP 项目的 JIRA 系统中已经记录了 12000 多个变更请求，其中体现的需求变更非常复杂，通过对需求变更技术债务的计算，可以辅助决策。

4.1　数据抓取

我们用时 5 天，共抓取了 2006 年 4 月到 2016 年 12 月的 12932 条变更请求报告。JIRA 系统中的变更请求有开放（open）、处理中（In progress）、重新开放（Reopened）、解决（Resolved）和关闭（Closed）5 种状态，为了能够计算出每个变更请求的时间花费，本文仅选择已修复的变更请求作为研究对象，即状态为解决（Resolved）和关闭（Closed）的变更请求，并选取了 9400 条数据。

4.2　需求变更债务值的计算

JIRA 中变更请求之间有 10 种关联关系，分别为：1)阻止关系，blocks 和 is blocked by；2)依赖关系，depends upon 和 is depended upon by；3)需要关系，requires 和 is required by；4)相关关系，relates to 和 is related to；5)重复关系，duplicates 和 is duplicated by；6)克隆（复制）关系，is cloned by 和 is a clone of；7)破坏关系，breaks 和 is broken by；8)包含关系，contains 和 is contained by；9)合并关系，incorporates 和 is part of；10)替代关系，supersedes。其中，符合影响类需求变更（*affect*）的关联关系是 is depended upon by，is required by 和 is related to；符合产生类需求变更（*produce*）的关系是 blocks 和 breaks；一般类需求变更请求之间没有关联关系。将符合需求变更的关联关系对应的 r 值设为 1，其余都为 0。经研究讨论，l 是一个逐渐加强的强度系数，将 l 设为 $l_i=1+0.02*(i-1)$。

利用第 3 节中的需求变更技术债务量化公式计算出每个需求变更的技术债务值，例如变更请求 HADOOP-4487 的债务计算如下：

$$RD=\{64887660+[(47734200+9593280+5105700+13250040+160418100+34489440+90000780+7058940+5616540)+(169212660+740700+6195240)\times 1.02]\}$$
$$=238.36$$

项目中其他需求变更技术债务的计算结果的排名(前20)如表1所列。

表1 需求变更技术债务的计算结果

Table 1 Calculation results of technical debt caused by requirement change

排名	编号(ID)	债务值(Debt)
1	HADOOP-4487	238.36
2	HADOOP-4998	195.16
3	HADOOP-5962	194.42
4	HADOOP-6581	167.07
5	HADOOP-4343	166.32
6	HADOOP-6589	159.95
7	HADOOP-6572	159.86
8	HADOOP-4656	156.98
9	HADOOP-5135	154.91
10	HADOOP-6201	141.20
11	HADOOP-7363	139.67
12	HADOOP-11745	136.27
13	HADOOP-6584	135.95
14	HADOOP-5081	135.40
15	HADOOP-6419	134.77
16	HADOOP-10854	134.68
17	HADOOP-4348	133.81
18	HADOOP-13073	132.36
19	HADOOP-4453	132.29
20	HADOOP-5219	132.29

4.3 实验结果与分析

实验结果的可视化可以更直观地反映出HADOOP项目中需求变更技术债务的情况。本文使用Gephi对实验结果进行可视化操作,如图4所示。需求变更技术债务值可以作为评估软件质量的重要指标,由图4可知,需求变更技术债务值最大的为HADOOP-4487。而需求变更技术债务值越大,就意味着需求变更完成的优先级越高。

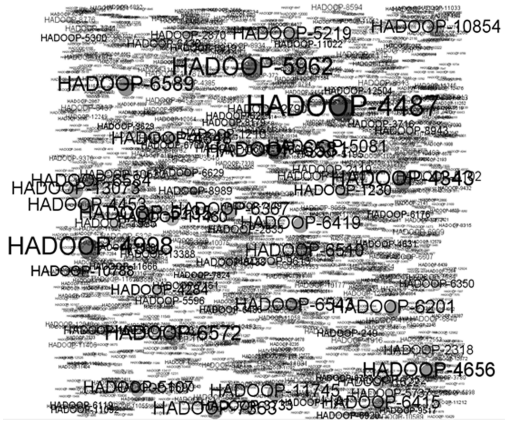

图4 需求变更技术债务值的可视化

Fig. 4 Visualization of technical debt value caused by requirement change

图5为HADOOP-4487需求变更技术债务增长情况,横坐标表示关联的其他变更请求(按提出时间排序),纵坐标表示债务值。开始阶段,债务值迅速增长,说明与其有关联的需求变更不断被提出,后来逐渐趋于平缓,最后债务被偿还。

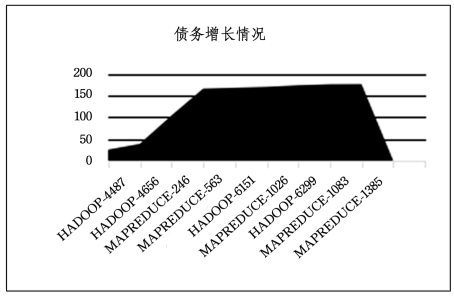

图5 需求变更技术债务的增长情况

Fig. 5 Growth of technical debt caused by requirement change

HADOOP-4487是第一个请求实现HADOOP安全机制的变更请求,标题为"Security features for Hadoop"。随后,安全方面的变更请求被不断提出,而HADOOP-4487变更请求也并没有被快速解决。直到一年后,Apache才专门成立了一个团队为HADOOP实现安全机制,偿还了安全需求方面的债务。HADOOP-4487属于产生类需求变更,通过分析HADOOP-4487需求变更技术债务值的增长情况,可以清晰地了解到:对于产生类需求变更,当债务值增长趋向平缓后,也即需求变更所涉及的方方面面都被考虑时,债务才被偿还。我们还跟踪到一个编号为HDFS-2006的变更请求,它希望HDFS能够提供一个功能来存储文件的扩展属性,但未受到重视。随后,一个编号为HADOOP-10150的变更请求被提出,它请求添加一个特性来保护HADOOP中的数据,而这个特性依赖于之前HDFS-2006的实现,因此开发人员返回来先实现HDFS-2006。变更请求HDFS-2006属于影响类变更请求,对于影响类需求变更,当受影响的需求变更数量到达最大值时,债务值不再积累,债务才能被偿还。而对于一般类变更请求,可以根据需要随时处理,偿还其债务。当出现一个新的变更请求时,债务值增长速度快并不一定是一件坏事。债务值越大,其影响的范围越大,越应该慎重考虑。

结束语 本文基于开源项目JIRA中的变更请求数据,对需求变更技术债务进行了研究,提出了需求变更技术债务的定义和量化,对项目中真实存在的需求变更技术债务进行分析,达到了预期的效果。本文工作还有很多方面需要深入研究与探讨,目前只将需求变更技术债务量化方法应用到了HADOOP项目组中,没有覆盖更多的平台和项目。大型开源项目HADOOP属于敏捷式开发,与传统模型相比也可能会有不同。下一步工作将尝试将此方法应用到其他实际项目中,如Bugzilla-Defect跟踪系统,或一些传统模型开发的项目中,以进一步验证量化方法的效果。技术债务是一个非常复杂的问题,本文仅采用时间指标来度量需求变更技术债务,还有较大的改进空间,而且技术债务也涉及到很多代码、维护环节,单独考查需求变更技术债务并不一定全面。下一步工作也将尝试一些其他度量指标,如代码行等,并且尝试对多种技术债务进行关联研究。

参 考 文 献

[1] ALVES N S R,MENDES T S,MENDONÇA M G D,et al. Identification and management of technical debt: A systematic

mapping study[J]. Information & Software Technology, 2016, 70: 100-121.

[2] LI Z, AVGERIOU P, LIANG P. A systematic mapping study on technical debt and its management[J]. Journal of Systems & Software, 2015, 101(C): 193-220.

[3] BUCKLEY J, MENS T, ZENGER M, et al. Towards a taxonomy of software change[J]. Journal of Software Maintenance & Evolution Research & Practice, 2010, 17(5): 309-332.

[4] MACCORMACK A, BROWN N, CAI Y, et al. Managing Technical Debt in Software-Reliant Systems[C]// Proceedings of the Fse/sdp Workshop on Future of Software Engineering Research-Foser. 2010: 47-52.

[5] CURTIS B, SAPPIDI J, SZYNKARSKI A. Estimating the size, cost, and types of technical debt[C]// Third International Workshop on Managing Technical Debt. IEEE, 2012: 49-53.

[6] LETIER E, STEFAN D, BARR E T. Uncertainty, risk, and information value in software requirements and architecture[C]// International Conference on Software Engineering. ACM, 2014: 883-894.

[7] ERNST N A. On the role of requirements in understanding and managing technical debt[C]// Third International Workshop on Managing Technical Debt. IEEE, 2012: 61-64.

[8] ABAD Z S H, RUHE G. Using real options to manage Technical Debt in Requirements Engineering[C]// Requirements Engineering Conference. IEEE, 2015: 230-235.

[9] GOKNIL A, KURTEV I, BERG K V D, et al. Change impact analysis for requirements: A meta modeling approach[J]. Information & Software Technology, 2014, 56(8): 950-972.

[10] HASSINE J, RILLING J, HEWITT J, et al. Change impact analysis for requirement evolution using use case maps[C]// Proceedings of the Eighth International Worksho o ib Principles of Software Evloution. IEEE, 2005: 81-90.

[11] AURUM A, WOHLIN C. The fundamental nature of requirements engineering activities as a decision-making process[J]. Information & Software Technology, 2003, 45(14): 945-954.

[12] BOEHM B W, SULLIVAN K J. Software economics: a roadmap [C]// CSE'00 Proceedings of the Conference on the Future of Software Engineering. ACM, 2000: 319-343.

[13] WHITE T. Hadoop: The Definitive Guide[M]. Farnham: O'Reilly Media, Inc. 2011.

[14] ORTU M, DESTEFANIS G, ADAMS B, et al. The JIRA Repository Dataset: Understanding Social Aspects of Software Development[C]// International Conference on Predictive MODELS and Data Analytics in Software Engineering. 2015: 1-4.

（上接第 64 页）

参 考 文 献

[1] AN B, MA J, CAO D, et al. Towards Efficient Resource Management in Virtual Clouds[C]// 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). Atlanta: IEEE Press, 2017: 320-324.

[2] CAO D G, AN B, SHI P C, et al. Providing Virtual Cloud for Special Purposes on Demand in JointCloud Computing Environment[J]. Journal of Computer Science and Technology, 2017, 32(2): 211-218.

[3] ZHU Y J, MA J M, AN B, et al. Monitoring and Billing of a Lightweight Cloud System Based on Linux Container[C]// 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). Atlanta: IEEE Press, 2017: 325-329.

[4] AN B, SHAN X D, CUI Z C, et al. Workspace as a Service: an Online Working Environment for Private Cloud[C]// 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE). San Francisco: IEEE Press, 2017: 19-27.

[5] KLUYVER T, RAGAN-KELLEY B, PÉREZ F, et al. Jupyter Notebooks-a publishing format for reproducible computational workflows[C]// International Conference on Electronic Publishing. 2016: 87-90.

[6] MCKINNEY W. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython[M]. America: O'Reilly Media, 2012: 111-150.

[7] MCKINNEY W. Pandas: a foundational Python library for data analysis and statistics[J/OL]. Python for High Performance and Scientific Computing, https://www.researchgate.net/publication/265194455_pandas_a_Foundational_Python_Library_for_Data_Analysis_and_Statistics.

[8] Infogram. Inforgram(Version1.0)[EB/OL]. https://www.infogram.com.

[9] DORY, MICHAEL, PARRISH A, et al. Introduction to Tornado: Modern Web Applications with Python[M]. America: O'Reilly Media, 2012: 67-97.

[10] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: Cluster computing with working sets[C]// Usenix Conference on Hot Topics in Cloud Computing. 2016: 10.

[11] GROPP W, THAKUR R, LUSK E. Using MPI-2: Advanced features of the message passing interface[M]. America: MIT Press, 1999: 42-55.

[12] Handsontable. Handsontable(Version1.0)[EB/OL]. https://www.handontable.com.

[13] XUE Z, LI R, ZHANG H, et al. DC-Top-k: A Novel Top-k Selecting Algorithm and Its Parallelization[C]// 2016 45th International Conference on Parallel Processing (ICPP). Philadelphia: IEEE Press, 2016: 370-379.

[14] HUNTER J D. Matplotlib: A 2D graphics environment[J]. Computing in Science & Engineering, 2007, 9(3): 90-95.

[15] VITTER J S. External memory algorithms and data structures: Dealing with massive data[J]. ACM Computing surveys (CsUR), 2001, 33(2): 209-271.

[16] YANG H, DASDAN A, HSIAO R L, et al. Map-reduce-merge: simplified relational data processing on large clusters[C]// Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. Beijing: ACM, 2007: 1029-1040.

[17] BORGERT S, MÜHLHÄUSER M. A S-BPM Suite for the Execution of Cross Company Subject Oriented Business Processes [C]// International Conference on Subject-Oriented Business Process Management. Eichstätt: Springer, 2014: 161-170.

[18] WANG H, SHI P, ZHANG Y. JointCloud: A Cross-Cloud Cooperation Architecture for Integrated Internet Service Customization[C]// International Conference on Distributed Computing Systems. IEEE, 2017: 1846-1855.