软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(10):3186−3205 [doi: 10.13328/j.cnki.jos.005795]

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

人机协作的用户故事场景提取与迭代演进*

王春晖[1,2,4], 金 芝[1,2], 赵海燕[1,2], 刘 璘[3], 张 伟[1,2], 崔牧原[1,2]

[1](北京大学 信息科学技术学院,北京 100871)

[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[3](清华大学 软件学院,北京 100084)

[4](内蒙古师范大学 计算机科学技术学院,内蒙古 呼和浩特 010022)

通讯作者: 金芝, E-mail: zhijin@pku.edu.cn

摘 要: 敏捷软件开发中常用用户故事表达需求:用户故事讲述了具体的用户角色希望软件完成的功能.系统新版本的需求常常来自多个用户故事的整合.随着版本的迭代,不断会有新的用户故事出现.用户故事的迭代式管理和整合是敏捷开发成功的关键.为帮助开发者掌握项目的用户故事需求,提出人机协作的用户故事理解、整合和管理方法,支持需求的迭代更新.具体而言,提出用户故事元模型和带场景细粒度描述的用户故事表示,从故事描述、功能特征属性以及场景 3 个维度表达用户故事.便于开发者理解用户故事所表达的功能需求.提出基于元模型的用户故事理解,以及用户故事表达元素的自动提取方法.提出人机协同式用户故事整合方法,使用功能场景图辅助开发者确定用户故事间的关联关系.提出基于功能场景图的用户故事迭代更新方法.案例研究展示了方法的可行性.

关键词: 敏捷需求工程;用户故事;需求抽取;需求迭代演进;用户故事关系挖掘

中图法分类号: TP311

中文引用格式: 王春晖,金芝,赵海燕,刘璘,张伟,崔牧原.人机协作的用户故事场景提取与迭代演进.软件学报,2019,30(10):3186−3205. http://www.jos.org.cn/1000-9825/5795.htm

英文引用格式: Wang CH, Jin Z, Zhao HY, Liu L, Zhang W, Cui MY, Human-assisted elicitation and evolution of user stories with scenarios. Ruan Jian Xue Bao/Journal of Software, 2019,30(10):3186−3205 (in Chinese). http://www.jos.org.cn/1000-9825/5795.htm

Human-assisted Elicitation and Evolution of User Stories with Scenarios

WANG Chun-Hui[1,2,4], JIN Zhi[1,2], ZHAO Hai-Yan[1,2], LIU Lin[3], ZHANG Wei[1,2], CUI Mu-Yuan[1,2]

[1](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[2](Key Laboratory of High Confidence Software Technology of Ministry of Education (Peking University), Beijing 100871, China)

[3](School of Software, Tsinghua University, Beijing 100084, China)

[4](College of Computer Science and Technolgoy, Inner Mongolia Normal University, Hohhot 010020, China)

Abstract: User stories are widely used in agile development projects. Every user story tells what the user/customer wants the system to do. However, a user story can only contain a small piece of the requirements but not the whole business logic. That means that when the customers submit user stories, the developers need to combine them together according to the relationships among them for producing or updating the system requirements. That is very tedious, time-consuming, and error-prone. This study proposes a human-machine

* 基金项目: 国家重点基础研究发展计划(973)(2015CB352201); 国家自然科学基金(61620106007, 61751210, 61272163, 61432020, 61690200)

Foundation item: National Basic Research Program of China (973) (2015CB352201); National Natural Science Foundation of China (61620106007, 61751210, 61272163, 61432020, 61690200)

本文由“面向 DevOps 的软件工程新技术”专题特约编辑荣国平、白晓颖、岳涛推荐.

收稿时间: 2018-09-01; 修改时间: 2018-10-31; 采用时间: 2018-12-14; jos 在线出版时间: 2019-05-22

CNKI 网络优先出版: 2019-05-22 15:26:00, http://kns.cnki.net/kcms/detail/11.2560.TP.20190522.1525.004.html

collaborative approach to support the user story understanding and system functional requirements generation. This approach proposes to include the scenarios as the fine-grained representation of user stories and presents a feature-scenario model to capture the elements of user stories in three dimensions, i.e., the story description, the function attributes, and the scenarios. It designs a three-step algorithm to accept the submission of user stories, extract the features of each user story, and construct its functional scenario. As there are relationships among different user stories, it defines three types of correlative relations among them based on the functional scenarios. With the help of the customers, it supports the measurement and the identification of these relations and then constructs the system's view of the functional requirements. It is also applicable when obtaining new user stories for tolerating the requirements evolution. A case study shows the feasibility of this approach.

Key words: agile requirements engineering; user story; requirements elicitation; requirement evolution; user story relation mining

软件开发过程中,需求很难一次性完整获得,而且也会频繁发生变更.传统方法试图在开发前完成并确认整个需求文档,但这样开发出的软件产品常常因不满足用户需要而使项目以失败告终[1].敏捷方法注重领域用户与开发团队的频繁交互,能够快速响应需求变化,在较短的时间内开发出满足用户需求的软件产品[2].但开发者如何能够准确理解领域用户需求并有效地维护和管理需求变更,仍然是困扰敏捷开发团队的难题.

一方面,在敏捷方法中,用户故事是领域用户表达需求的最主要方式,该种方式只对意图进行简短说明[3],需要开发者与领域用户进一步沟通,以形成系统功能说明.另一方面,在发布规划时[3](规划迭代开发选择的用户故事,确定项目时间表),需要根据领域用户对特定功能特征的急切程度以及用户故事间的关系排列用户故事的优先级.目前,一般通过领域用户与开发者之间的反复交流和讨论,建立起用户故事索引表,人工梳理故事间的关系[1−3].大型软件开发一般会产生大量用户故事,每天都出现大量的新增需求,比如百度公司每天新增需求卡片近 6 700 个[4].用户故事间的关系可能错综复杂,它们常常重复和不一致.建立起完整的故事关系,特别是建立新增的和已有的用户故事间的关系非常困难.如果能有一种(半)自动化的手段帮助处理用户故事文档,进行需求抽取以及关联关系挖掘,将有可能辅助实现高效的用户故事管理.

为了降低敏捷团队成员间沟通和理解的困难程度,领域用户可以描述更多的系统行为,以方便开发者确定系统功能.行为驱动的敏捷方法[5,6]提出使用场景(scenario)更细粒度的描述用户故事,表达用户故事在不同状况下的系统活动,便于在开发前进行需求验证及测试.场景描述了用户故事在特定上下文中的结果.这种描述方法以领域用户所希望的系统行为出发,引导领域用户提出需求.系统行为源于待开发软件产品的业务结果,表明,为了达成业务结果应该如何去做.因而,容易被领域用户接受.同时,在场景中关注目标系统的细粒度行为需求,更容易被开发者理解和实现.

一个软件系统通常有很多用户故事,需要一种有效的方式将这些用户故事组织在一起,建立用户故事之间的关系,形成对未来系统的整体需求[7,8].目前已有一些整合用户故事的方法[9−11],它们通过建立用户故事与需求模型之间的映射,实现从一组用户故事到需求模型的转换,包括从用户故事到 Use-Case 模型[12]、业务流模型[10]以及目标模型[11]等的转换.也有方法借助自然语言分析技术自动地从用户故事中提取概念模型,并检查需求的完整性[9,13,14].但是由于自然语言描述本身的二义性及用户故事描述的不规范,全自动的模型转换非常困难.目前,大多数方法都仅提供模型编辑工具,辅助模型生成[9−12]等.

敏捷开发过程中,需求并不是事先定义好的,而是通过利益相关者之间的频繁交互,随着迭代开发过程逐步变得清晰[1−3,15,16].在每个迭代周期结束时,领域用户根据对已实现功能的测试及评估结果提出需求变更申请,主要包括添加或删除功能或修改已实现的功能.变更的功能(用户故事)应该快速地融合到系统需求模型中,以实现模型的快速更新,还需要检查更新的需求与前期版本需求是否存在冲突[17].目前,用户故事管理的办法主要是建立用户故事列表和索引卡,人工分析用户故事的变化[3]以及变更的成本和风险.有的研究通过分析多个连续版本的需求文档来评估需求变更的风险[18,19],也有的研究提供辅助平台,借助相关领域知识帮助管理需求演进[20].

本文提出一种敏捷需求工程场景下的人机协作式用户故事理解和系统功能需求获取方法.该方法提出带场景的用户故事表示,当领域用户提交一组用户故事后,可根据用户故事元模型,借助用户故事模板以及句法规则,采用自然语言分析技术,进行用户故事的特征抽取和功能表达.提出基于用户故事概念关系推理规则的用户

故事关系挖掘,识别用户故事间的合作关系、执行依赖关系以及近义关系.经过领域用户的确认,获得用户故事间的关系,生成系统功能场景.出现用户故事更新请求时,可借助系统功能场景,交互式地更新系统需求.案例研究展示了方法的可行性.

本文第 1 节介绍方法的整体框架.第 2 节介绍用户故事和系统功能场景的概念模型.第 3 节介绍用户故事场景抽取.第 4 节介绍用户故事关系挖掘及冲突检查,以及系统功能场景的生成与更新.第 5 节通过一个含有 24 个用户故事、57 个场景的在线购物系统实验案例,验证本文方法的可行性.第 6 节介绍相关工作.最后是结语和展望.

1 整体框架

人机协作的系统功能需求管理方法,主要包括用户故事功能场景抽取、系统功能场景构建以及用户故事场景更新 3 部分工作,整体工作流程框架如图 1 所示.

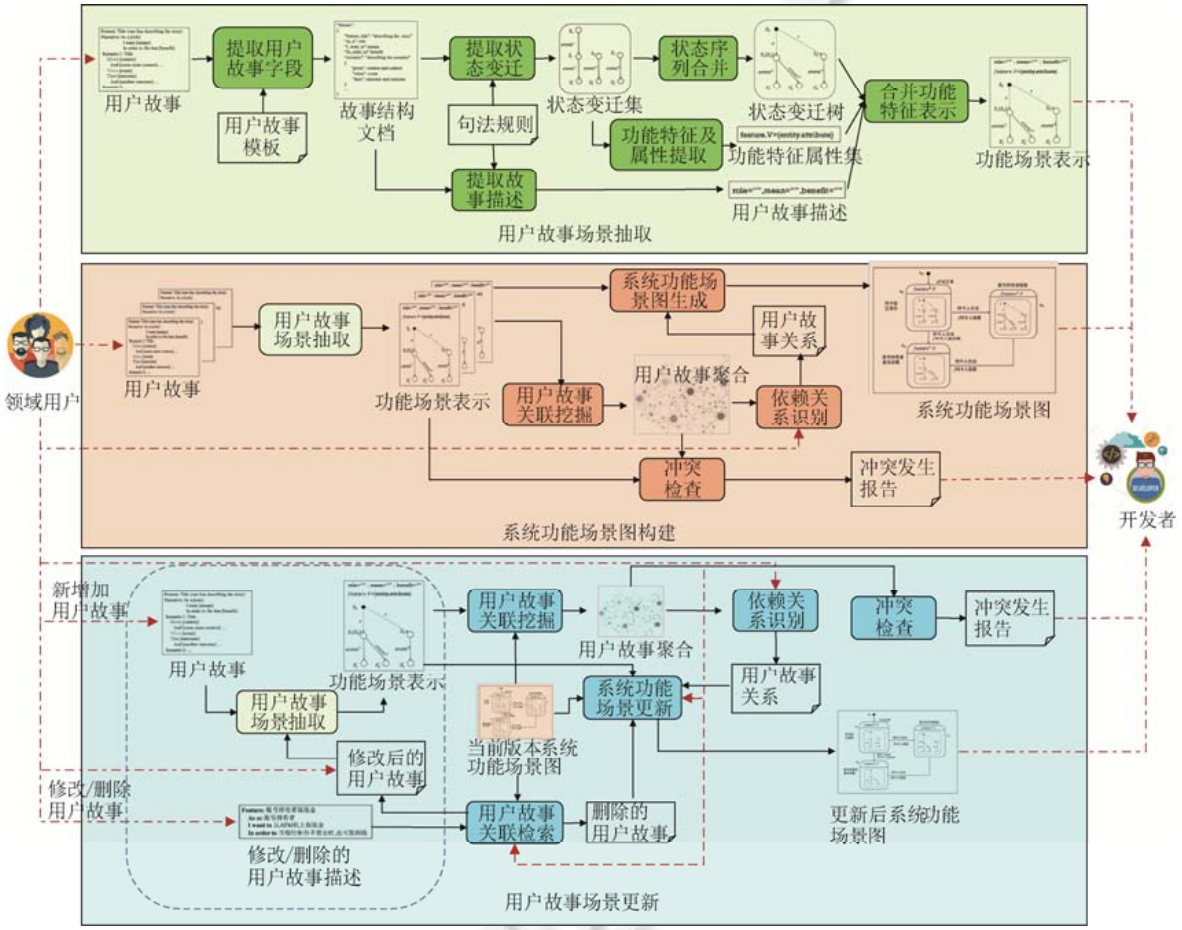


Fig.1 The framework for elicitation and evolution of user stories with scenarios

图 1 用户故事场景抽取与迭代演进整体框架

用户故事功能场景抽取以带有场景的用户故事为输入,使用自然语言分析方法,抽取功能场景的结构化模型表示,得到功能场景表示.首先,借助用户故事模板提取带场景的用户故事的字段信息,得到带故事语义标签的结构化文档,称为故事结构文档.然后,应用抽取规则,从故事结构文档中抽取功能场景表示,包括故事描述、场景以及功能特征属性集 3 个部分.其中,故事描述部分记录了故事名字以及角色、意图等.从故事结

构文档的对应存储字段抽取.场景部分描述故事的意图达成,表达为在具体的上下文(即用一组业务实体属性的取值表示的状态集)中,当某个事件发生时,意图达成的结果(即新的状态集).这样,场景实际上就是从一个状态集出发,经过事件,达到结果状态集,这称为一个状态变迁.一个用户故事的意图是否达成,可能有多种情况.所以,一个用户故事可能有多个场景.一个用户故事中的所有场景组成一个状态变迁集.应用状态合并规则,合并状态变迁集,生成状态变迁图.从状态变迁集中提取反映意图达成所需要的系统功能相关的业务实体属性,得到功能特征属性集.故事描述、功能特征属性集以及状态变迁图组合在一起,形成一个用户故事的功能场景表示.

系统功能场景构建以一组带场景的用户故事为输入,通过功能场景的整合,形成系统用户故事需求.首先,从用户故事集中抽取用户故事功能场景,得到具有故事描述、功能特征属性集及状态变迁图的功能场景表示,称为功能特征节点.这样,用户故事集就表达为一组功能特征节点.其次,通过和领域用户交互确定用户故事之间的关系,分为合作关系、执行依赖关系以及近义关系 3 类,通过相应的推理规则计算用户故事间关系的关联强度,将存在较强关联的用户故事对作为候选关系集推荐给领域用户.领域用户确认后,得到用户故事间的关系集.对存在近义关系的用户故事要进行冲突检查(语义冗余或不一致).在没有冲突发生的情况下,用户故事间的关系将功能特征节点连接起来,形成系统功能场景.

用户故事场景更新以用户提交的更新和当前版本的系统功能场景为输入,根据不同的更新请求,交互式地进行需求的更新,得到系统功能场景的更新.更新操作包括添加新用户故事、删除和修改已有用户故事.增加新用户故事需要领域用户提供一个带场景的故事,通过执行用户故事场景抽取创建新的功能特征节点,在当前版本的系统功能场景中寻找可能与它相关的片段,进行依赖关系识别,确定新用户故事在当前版本系统功能场景中的位置及关联关系,这个过程也需要领域用户的参与.然后进行冲突检查,如果没有冲突,则插入新故事.删除用户故事首先需要领域用户输入要删除用户故事的关键信息,然后在当前版本系统功能场景中进行用户故事检索,以确定要删除的故事及其关系.最后删除该用户故事并修正其他用户故事之间的关系.修改已有用户故事首先要在当前版本系统功能场景中检索用户故事定位要更新部分,然后修改用户故事文档,得到修改的用户故事.最后,再将这个用户故事作为新的故事文档添加到系统功能场景中.

图 1 给出的方法整体框架,蕴含着以下 3 个方面的问题.

(1) 系统功能场景的表示.系统功能场景刻画了带场景的用户故事的结构化表示以及用户故事之间的关系.系统功能场景实际上是系统的功能模型,基于系统功能场景的用户故事管理是一种模型驱动的管理,它直接制导了用户故事功能抽取、整合和更新.第 2 节将对此给予重点介绍.

(2) 用户故事场景抽取.这是指根据带场景的用户故事构造系统功能场景中的功能特征节点,具有故事描述、功能特征属性集以及状态变迁图 3 个属性.它是生成系统功能场景和实现更新管理的基础.本文借助自然语言分析技术和故事特征句法规则,从用户故事文档中构建功能特征节点.这方面的内容将在第 3 节加以介绍.

(3) 系统功能场景的生成与更新.建立用户故事间的关系是生成和管理系统功能场景的关键.由于用户故事都是独立提交的,其中并没有显式表达和其他用户故事的关系,如何辅助领域用户快速识别用户故事间的关联,并检查用户故事之间的冲突,成为关键.第 4 节将主要讨论这一问题.

2 用户故事及系统功能场景的概念模型

领域用户采用用户故事描述他们对软件系统的需求.讲述作为一个特定角色的使用者,需要什么功能,完成什么操作,带来什么好处[3].为了对用户故事进行细粒度的分析,提取其中的功能需求,可以进一步引入场景描述.即用户故事包括至少一个场景,表达用户故事发生的实例.另一方面,系统功能场景则站在未来软件系统的角度表达系统应该具有的功能及完成的操作.本节通过 ATM 机系统的案例介绍带场景的用户故事描述以及系统功能场景的概念模型.

2.1 带场景的用户故事

带场景的用户故事包括故事描述及场景.故事描述包括故事名、角色、意图以及收益.场景包括场景名、

上下文、事件以及结果.一个用户故事可以包括一个或多个场景.其描述模板及其概念元素层次关系如图 2 所示.

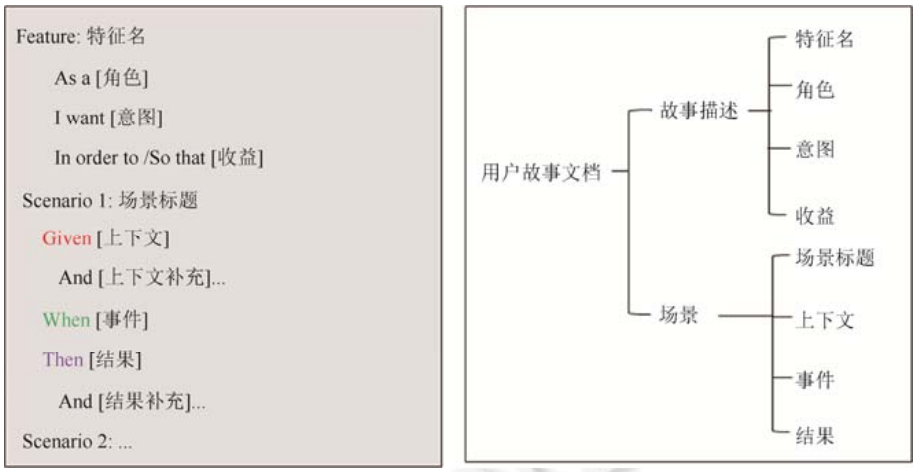


Fig.2 The template and the hierarchical relation of elements in user-story description document

图 2 带场景的用户故事描述模板及其概念元素层次关系

故事名一般以短句概括用户故事的功能.角色表示谁将使用这个功能,意图表示角色希望系统提供什么样的功能,收益表达了系统实现该功能后的好处.场景描述了用户故事在特定上下文中,发生某个事件而产生的结果.上下文和结果一般描述与功能相关的业务实体(属性)的某种取值,体现了功能业务的状态特性,用一个描述状态(实体-属性-值)的短句表达,分别用关键字 Given 和 Then 连接.当上下文或结果涉及多个状态表达时,将每个状态描述短句分开,并用 and 连接.上下文和结果之间的切换是由于发生某个事件(如动作或条件达成),事件用关键词 When 连接.

表 1 描述了持卡人在 ATM 机上取现金的用户故事.该用户故事分 3 个场景描述意图是否达成:成功取款、卡余额不足不能取款和卡无效无法取款.取现金业务与账号余额、卡是否验证通过(是否是合法用户)、取款金额以及 ATM 机余额等实体属性相关,称为与用户故事功能(意图)相关的功能特征属性集.这些功能特征属性在每个场景中实例化(取值),表示这些功能特征属性处于“上下文”状态下(前置状态),由于某个事件发生,功能特征属性处于“结果”状态(后置状态).如 scenario1 前置状态中描述的短句含义是:在账号的余额是 balance,卡被验证通过且 ATM 机有足够的现金.事件是“持卡人取现金 cash”.后置状态描述的短句含义是:ATM 吐出现金,系统提示取款成功,账号余额会修改.

Table 1 The user story for card holder withdrawn money

表 1 “持卡人取现金”用户故事描述文档

Feature: 持卡人取现金 **As** an 持卡人 **I want to** 从 ATM 机上取现金 **In order to** 当银行柜台不营业时,也可取到钱		
Scenario 1: 成功取款 **Given** 账号的余额是 balance **And** 卡被验证通过 **And** ATM 机的现金储备是足够的 **When** 持卡人取现金 cash **Then** ATM 的吐币金额是 cash **And** 系统提示信息是取款成功 **And** 账号的余额是 balance−cash	**Scenario 2**: 卡余额不足 **Given** 账号的余额是 balance **And** 卡被验证通过 **And** ATM 机的现金储备是足够的 **When** 持卡人取现金 cash **Then** ATM 的吐币金额是 0 **And** 系统提示信息是取款不成功 **And** 账号的余额是 balance	**Scenario 3**: 卡无效 **Given** 卡没有被验证通过 **When** 持卡人取现金 cash **Then** ATM 的吐币金额是 0 **And** 系统提示信息是取款不成功 **And** 卡被收回

为了让开发者能够充分理解用户故事中每段描述的含义.建议在每个关键词后面使用规范的句型,见表 2.

Table 2 Recommended pattern for a sentence followed by a key word in a user story

表 2 用户故事场景自然语言描述关键词后面对应短句的推荐描述句型

关键词	句型
feature	主+(状语)+谓+(定语)+宾语
As a	名词短语,名词短语
I want to	(状语)+动词+(定语)+宾语
In order to	–
Scenario	主+(状语)+谓+(定语)+宾语
Given/and/then	名词+(“的”+名词)+“是/不是”+名词(形容词) 名词+(“的”+“名词”)+(没有)+“处于”+名词+“状态” 名词+(“的”+“名词”)+(没有)+“被”+动词+(补语)
when	主+(状语)+谓+(定语)+宾语,名词+(“的”+名词)+“是”+名词; 名词+(“的”+“名词”)+“处于”+名词+“状态”

2.2 系统功能场景

系统功能场景是一个由功能特征节点和关系组成的图.系统功能场景的概念模型如图 3 所示.

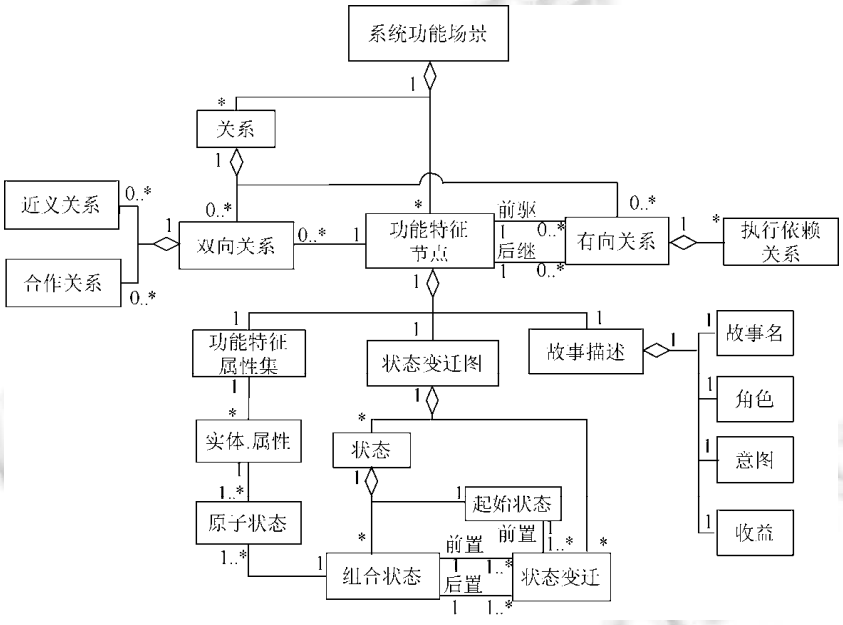


Fig.3 The conceptual model of system feature-scenario

图 3 系统功能场景的概念模型

每个功能特征节点有 3 部分信息:故事描述、功能特征属性集以及状态变迁图.故事描述包括故事名、角色、意图以及收益.功能特征属性集是一个由实体-属性组成的集合.状态变迁图是多个场景的状态变迁组织在一起的图结构,每个状态变迁从一个状态转换到另一个状态.其中,起始状态是状态图的开始点,可以连接一个或多个组合状态.组合状态可以作为前置状态或后置状态,通过状态变迁关联起来.组合状态中的每个原子状态是功能特征属性集中某个实体-属性的实例(取某个值).

功能特征节点之间存在 3 种关系:合作关系、执行依赖关系和近义关系.合作关系表达一组故事通过合作共同达成一个较大规模的故事.执行依赖关系表明一个故事的执行依赖于另一个故事的实现.近义关系表明故事可能具有相似的语义.合作关系和近义关系是双向关系(互为合作关系或近义关系),一个功能特征节点可以没有双向关系或有多个双向关系.执行依赖关系是有向关系(一个功能特征节点 X 是另一个功能特征节点 Y 的前驱,Y 是 X 的后继),一个功能特征节点可以不作为其他任何功能特征节点的前驱(后继),也可以作为多个功能

特征节点的前驱(后继).

设实体、属性、值、事件是系统的 4 个基本数据集,分别表示为 *Entity*、*Attribute*、*Value*、*Event*.实体具有名字及属性,属性包括名字及类型,事件描述动作或者某个条件(如系统所处状态)的发生.值是满足属性类型的数据.对于图 3 所示概念模型中的元素,分别给出具体定义和形式化表示.

定义 1. 原子状态 S 是一个三元组(*entity*,*attribute*,*value*),其中,$entity \in Entity$ 表示实体,$attribute \in Attribute$ 表示实体的属性,$value \in Value$ 表示属性的取值,也可以表达为 *entity.attribute*=*value*,多个状态形成一个状态的集合 *states*.

定义 2. 组合状态 $S=\{s|s \in States\}$,是一个原子状态的集合.从场景上下文中提取的状态称为前置状态(*preState*),从结果中提取的状态称为后置状态(*preState*).

例如,表 1 中 *scenario*1 分别提取前置状态 S 和后置状态 S',S={(账号.余额=*balance*),(卡.状态=验证通过),(*ATM*.现金储备=*enough*)}(当描述状态的短句中省略了属性名词时,则认为是某实体的"状态").S'={(账号.余额=*balance*–*cash*),(卡.状态=验证通过),(*ATM*.现金储备=*enough*)},(*ATM*.吐币金额=*cash*)(系统.提示信息=取款成功).

定义 3. 状态变迁 t 是一个三元组(S,*event*,S'),其中,S 是前置状态,S'是后置状态,$event \in Event$ 表示事件.表示从组合状态 S 通过事件 *event* 变换到组合状态 S'.多个状态变迁组成了一个状态变迁的集合,表示为 T.

状态变迁描述了从一个前置状态通过事件发生变化到后置状态的过程.一个变迁的目标组合状态可能是另一个状态变迁的源组合状态,形成一个状态序列,表示为 $(S_1,event^1,S'_1,event^2,S'_2,...)$.

定义 4. 属性集 $V=\{v_i|i=1,2,\ldots,K\}$,是由 K 个实体-属性 v 组成的集合,每个 v 是一个二元组(*entity*,*attribute*),表示一个实体的属性,该属性在不同的状态会取不同的值.记为 *entity.attribute*.

例如,表 1 的 *scenario*1,S 的属性集 V={账号.余额,卡.状态,*ATM*.现金储备},S'的属性集 V'={账号.余额,卡.状态,*ATM*.现金储备,*ATM*.吐币金额,系统.提示信息}.

定义 5. 场景属性集 $scenario_i.V=V \cup V'$(其中,V 是场景 $scenario_i$ 的前置状态属性集,V'是场景 $scenario_i$ 的后置状态属性集).

例如表 1 中场景 *scenario*1 的属性集 *scenario*1.$V=V \cup V'$={账号.余额,卡.状态,*ATM*.现金储备,*ATM*.吐币金额,系统.提示信息}.

场景状态补齐:对照场景的属性集,S 和 S'可能缺少个别属性的状态信息,例如 S 中没有描述(*ATM*.吐币金额)和(系统.提示信息).为了保持一致,向 S 中添加这两个属性,并且赋一个 NULL 值,表示对应的实体-属性取值不确定.如 S 补充属性和值后变为 S={(账号.余额=*balance*),(卡.状态=验证通过),(*ATM*.现金储备=*enough*),(*ATM*.吐币金额=*NULL*),(系统.提示信息=*NULL*)}.

定义 6. 功能特征属性集 $feature.V=\cup_{i=1}^{N} scenario_i.V (i=1,2,...,N)$, N 表示一个用户故事中场景的个数.

状态变迁图生成:假设表 1 中每个场景的前置状态用 S_i 表示,而后置状态用 S'_i 表示,事件用 $event^i$ 表示,则状态序列集可表示为 $\{(S_1,event^1,S'_1),(S_2,event^2,S'_2),(S_3,event^3,S'_3)\}$.执行场景状态补齐,6 个组合状态如图 4 左侧部分所示.如果场景中的组合状态用点表示,状态之间的事件作为一条有标记的边,合并相同状态,并添加一个起始状态 S_0 和空转移的边 ε,连接每个状态变迁的初始状态,这样,状态变迁集表达为一个连通的状态变迁图,如图 4 右侧部分所示.

故事描述、功能特征属性和状态变迁图是一个功能特征节点的 3 部分.通过关系连接起来形成一个系统功能场景.

定义 7. 一个功能特征节点是一个三元组(U,V,G),U 也是一个四元组(*title*,*role*,*mean*,*benefit*)表示节点的用户故事,V 表示功能特征属性集,G 表示状态变迁图.

定义 8. 系统功能场景是一个图.用二元组(N,E)表示.

- N 是节点的集合,系统功能场景中的节点 $n \in N$,是一个功能特征节点.
- E 是一个故事关系集合,系统功能场景的边 $edge \in E$,是一个三元组(n_1,e,n_2),表示节点 n_1 到目标节点 n_2 存

在关系 *e*.关系 *e* 有 3 种类型:合作关系、执行依赖关系和近义关系.执行依赖关系有方向,而另外两种关系没有方向,表示两个故事互相具有此种关系.

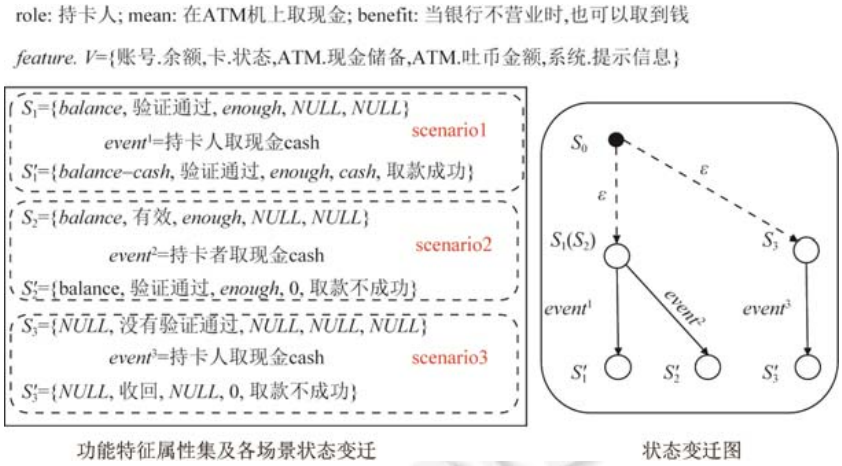


Fig.4 Function-scenario description of the cases in Table 1

图 4 表 1 案例功能场景表示

2.3 系统功能场景举例

写用户故事一般从复杂用户故事开始,即领域客户先写一个“大”用户故事,表达相对宏观的需求.然后通过分解用户故事逐步使故事具体化,形成一些小故事[3].例如,ATM 自动取款机办理业务是一个复杂的故事.表 1 的例子讲述了作为一个持卡人,在 ATM 机上取款的故事.假设还包括另外两个故事:持卡人验证身份,持卡人查询余额,详见表 3(省略了场景中的细节).

Table 3 The description of two sub user stories for ATM business

表 3 ATM 自动取款机业务的两个子故事

Feature 持卡人验证身份	**Feature** 持卡人查询余额
As a 持卡人	**As a** 持卡人
I want to 验证持卡人身份	**I want to** 查询账号余额
In order to 当身份验证成功后,可以办理查询余额、取款等业务	**In order to** 当知道账号余额,可以取现金
Scenario 1 身份验证通过	**Scenario 1** 成功查询到余额
…	…
Scenario 2 身份验证不通过	**Scenario 2** 卡无效
…	…

持卡人取款、持卡人验证身份和持卡人查询余额,这 3 个用户故事具有相同的角色,都涉及账号、卡状态、ATM 机等领域概念,而且在 In order to 描述字段中隐含了用户故事之间的依赖关系.根据这些概念之间的联系,可以发现这 3 个用户故事之间的关系,得到后文图 5 所示的系统功能场景图.

3 用户故事场景抽取

用户故事场景抽取以一个带场景的用户故事为输入,通过故事结构文档生成、状态变迁提取、状态变迁图生成,将用户故事表达为功能场景表示.本节介绍用户故事场景抽取生成功能场景表示的过程.

3.1 故事结构文档生成

带场景的用户故事是一个有标签的文档,每个标签后面的信息表达特定的语义,满足表 4 用户故事文档描述规约.

按照表 4 规约从用户故事提取字段信息,存放在故事结构文档.后文图 6 所示为表 1 中用户故事的结构文档表示.

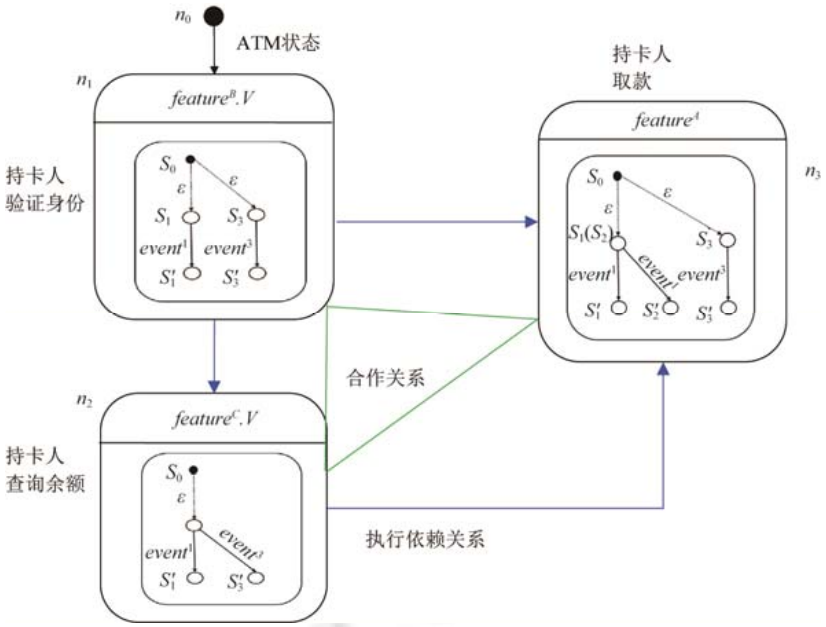


Fig.5 An example of a feature-scenario

图 5 系统功能场景示例

Table 4 The description rule of a user story

表 4 用户故事文档描述规约

〈特征定义〉::=feature-〈标题〉
〈用户故事〉::=as a 〈角色〉, I want to〈意图〉,In order to〈收益〉
〈场景定义〉::=scenario-〈编号〉〈标题〉
〈前置状态〉::=Given〈描述状态的短句〉〈连接符〉〈描述状态的短句〉
〈事件〉::=When〈条件|动作〉
〈后置状态〉::=Then〈描述状态的短句〉〈连接符〉〈描述状态的短句〉
〈连接符〉::=And
〈状态〉::=〈实体名〉.〈属性名〉.〈值〉
〈条件〉::=〈描述状态的短句〉
〈动作〉::=〈角色名〉〈动词〉〈操作对象〉
〈值〉::==〈表达式〉|〈条件表达式〉

3.2 状态抽取与状态变迁图生成

为了生成状态变迁图以及提取功能特征属性集,状态(实体-属性-值)首先从故事结构文档中“state”标签后面的短句中提取.使用自然语言分析工具(HanLP[21]),对状态描述语句进行分词、词性标注和依存句法分析,然后与句法模板(表 2)匹配,根据匹配句型找出实体、属性和值.例如,“账号的余额是 balance”与“名+名+是+名”状态模式匹配,进而找出状态{账号.余额=balance}.

从用户故事中识别所有状态,并根据状态表达语句的位置,分别存储在 preState 和 postState 这两个集合中.然后进行场景状态补齐,得到统一的实体-属性集,具体步骤如下.

• Step 1:从 preState 和 postState 中的每个状态中提取实体-属性,存储到属性集 V 和 V'中.

• Step 2:取属性集 V 和 V'的并集,得到一个关于这个场景的属性集 $scenario.V=V\cup V'$.

• Step 3:在 preState 中补充实体-属性 $scenario.V-V$,并给 $scenario.V-V$ 的实体-属性赋 NULL 值.在 postState 中补充实体-属性 $scenario.V-V'$,并给 $scenario.V-V'$的实体-属性赋 NULL 值.

通过场景状态补齐,preState 和 postState 具有相同场景属性集.提取 event 字段信息,组成一个状态变迁序列(preState,event,postState).

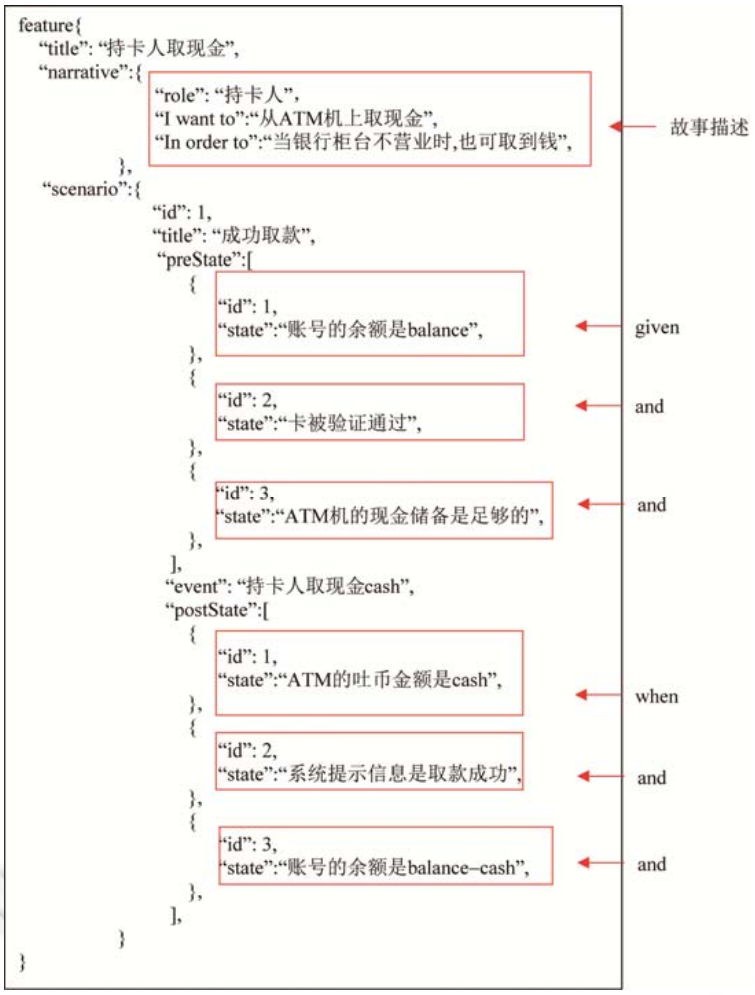


Fig.6 The json document of a user story in Table 1

图 6 表 1 故事结构文档 json 文件(部分)

上述过程应用于用户故事的每一个场景,最终得到一个状态变迁序列的集合.合并状态变迁序列集合中相同的组合状态(组合状态 State 1 和 State 2 具有相同状态数,State 1 中的所有状态在 State 2 中都能找到,且 State 2 中的所有状态在 State 1 中都能找到),然后添加初始状态和转换边,生成状态变迁图.状态变迁图生成过程如下.

• Step 1:提取每个场景的状态变迁序列,得到一个状态变迁序列集合 T.

• Step 2:对 T 中所有场景执行场景状态补齐,得到功能特征属性集 $feature.V$.

• Step 3:匹配 T 中不同场景的组合状态,如果有相同的组合状态,则持行 Step 4,否则,执行 Step 5.

• Step 4:根据合并规则(见表 5)合并状态变迁序列.如果没有违反合并规则,则转 Step 5,否则,报告错误.

• Step 5:增加一个空的状态节点,作为起始节点.连接起始节点与合并后状态序列集中所有没有前驱的组合状态节点,并在标记事件的边上添加ε.形成一个状态变迁图.

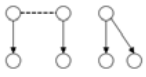
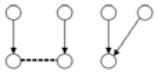
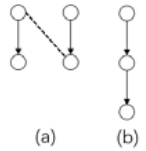
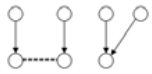
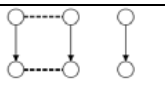
状态变迁图表达了用户故事在不同上下文情况下意图是否达成的状态结构化组织方式.状态变迁图是一个有向无环图,从起始点出发,可以遍历所有状态.使用状态变迁图可以帮助开发者了解与功能实现相关的业务流,同时也便于开发和测试.

表 5 列举了状态变迁序列合并的 6 种情况,其中,(a)表示合并之前两个状态序列的对应关系,(b)表示合并之后的情况.虚线表示两端的节点状态相同,能够合并在一起.这 6 种情况中,1~3 能够正常自动合并,4~6 可能存在

语义表达冲突,不能进行合并.

Table 5 The rules for merging states sequence

表 5 状态变迁序列合并规则

序号	对应关系	说明	图示	序号	对应关系	说明	图示
1	两个状态变迁序列的 preState 相同	不同事件产生了两种不同的结果	 (a) (b)	4	两个状态变迁序列的 postState 相同	在不同的上下文执行相同的事件达到同样的结果,说明上下文中有冗余的状态	 (a) (b)
2	一个状态变迁序列的 preState 与另一个状态变迁序列的 postState 相同	一个场景的目标状态是另一个场景的起始状态	 (a) (b)	5	两个状态变迁序列的 preState 和 postState 逆向相同	存在矛盾	 (a) (b)
3	两个状态变迁序列的 postState 相同	在不同的上下文执行不同的事件达到同样的结果	 (a) (b)	6	两个状态变迁序列的 preState 和 postState 分别对应相同	存在重复	 (a) (b)

3.3 生成功能场景表示

功能场景表示除了包含功能场景属性集和状态变迁图外,还包括故事描述.故事描述由用户故事名、角色、意图及收益 4 部分信息组成,分别从故事特征字段文档的“title”和“narrative”字段中提取.结合功能属性特征集提取和状态变迁图生成,生成功能场景表示如算法 1 所示.

算法 1. 功能场景表示生成.

输入:一个用户故事的故事结构文档 *fileInput.json*;

输出:一个功能场景表示 *fsNode*.

伪代码:

1. *name*=*getTitle*(*fileInput*)
2. *narrative*=*getNarrative*(*fileInput*)
3. *role*=*getRole*(*narrative*),*mean*=*getMean*(*narrative*),*benefit*=*getBenefit*(*narrative*)
4. *U*=*set*(*name*,*role*,*mean*,*benefit*)//将角色、意图及收益存放在故事描述中
5. $V=\varnothing$ //存放功能特征节点的属性集
6. $C=\varnothing$ //一个初始状态集
7. *scenario*=*getScenario*(*fileInput*)
8. for *i*=1 to *scenario.size*//对文档 *di* 中 *K* 个场景描述
9. $S=\varnothing$
10. *preState*=*getpreState*(*scenario*[*i*])
11. for *j*=1 to *preState.size*
12. $S=S\cup extracState(preState[j].state)$ //从状态描述语句中抽取状态
13. *V*=*extractAttributeSet*(*S*)//从状态集合中提取属性集
14. *e*=*getEvent*(*scenario*[*i*].*event*)
15. $S'=\varnothing$
16. *postState*=*getpostState*(*scenario*[*i*])
17. for *j*=1 to *postState.size*
18. $S'=S'\cup extractState(postState[j].state)$
19. $V=V\cup extractAttributeSet(S')$
20. $C=C\cup getChange(S,e,S')$ *C*//添加一个状态序列
21. *C*=*mergeStateSequence*(*C*)//合并状态序列,参照表 5 合并规则

22. *G*=*createScenarioGraph*(*C*)//创建场景树
23. *fsNode*=*createNode*(*U*,*V*,*G*)
24. return *fsNode*

4 系统功能场景生成与更新

系统功能场景包括一组功能特征节点和它们之间的关系,以系统的视角呈现用户故事需求.本节介绍用户故事关联关系挖掘方法、多个用户故事之间的冲突检测以及系统功能场景生成与更新.

4.1 用户故事关系挖掘

用户故事之间可能存在的关联语义体现在图 7 所示的用户故事概念当中(源自于图 2).图中虚线框表示在图 2 用户故事概念的基础上细化(分解成分),增加功能场景属性集,灰色底纹部分表示暂时不考虑的概念.

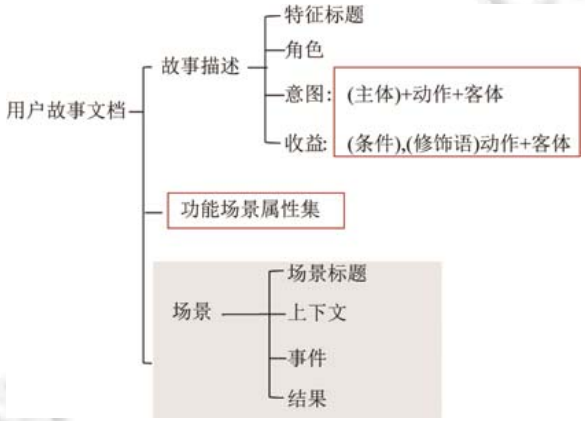


Fig.7 The hierarchical relation of concepts in a user story for constructing relationship

图 7 用户故事关系概念元素层次关系

用户故事 u_1,u_2 之间的关系包括合作关系、执行依赖关系以及近义关系.合作关系表达一组故事通过合作共同完成一个较大规模的故事.这些故事一般具有相同的角色和客体(功能的操作对象),功能特征属性集有重叠.执行依赖关系表明一个故事的执行依赖于另一个故事的实现.这些故事一般在功能描述上有相关性,如在“收益”中描述功能依赖.近义关系表明故事具有相似的含义,表现在用户故事概念模型的各部分都很相似.

1) 合作关系关联强度计算

合作关系关联强度从角色维度、功能客体(意图中客体)和功能特征属性相关度这 3 个方面加以度量.假设 u_1,u_2 分别表示两个用户故事,r_1,r_2 分别表示这两个故事的角色,o_1,o_2 分别表示这两个故事意图中的客体,V_1,V_2 分别表示这两个故事的功能特征属性,则用户故事 u_1,u_2 的合作关系强度 $S_{co}(u_1,u_2)$等于角色相似度、功能客体相似度与共同功能特征属性比例的加权平均,计算公式如下:

$$S_{co}(u_1,u_2)=\alpha_1 sim(r_1,r_2)+\alpha_2 sim(o_1,o_2)+\alpha_3\frac{|V_1\cap V_2|}{|V_1\cup V_2|} \tag{1}$$

本文综合使用自然语言分析工具中基于词向量的相似度计算方法[22]和语义相似度[21]的方法计算两个单词的相似度.$\alpha_1+\alpha_2+\alpha_3=1$,$S_{co}(u_1,u_2)$是一个 0-1 之间的小数.

2) 执行依赖关系关联强度计算

执行依赖关系关联强度与功能特征属性和“收益”中描述的功能(动作+客体)是否关联到另外一个用户的功能(“意图”中动作+客体)相关.假设 f_1,f_2 分别表示故事 u_1,u_2 的功能,其中,f_i=(*actionverb*,*object*)($i\in\{0,1\}$).分别从“意图”的动作和客体中获取.cf_1,cf_2 分别表示收益的描述中提到的功能,同样表达为 cf_i=(*actionverb*,*object*($i\in\{0,1\}$)).需要在 f_i 和 cf_i 之间做一个映射,计算它们的关联强度,称为功能相关度.举例如下:如第 2.3 节中“持卡人

查询余额”的用户故事中“收益”字段描述了“当知道账号余额,可以取现金”,而“持卡人取现金”的用户故事中的“意图”描述了“在 ATM 机上取现金”.借助自然语言分析方法(分词、词性标注、依存文法分析[23])可以得到 cf_1=(取,现金),f_2=(取,现金),它们的功能相关度为 1.

功能相关度与动词及客体相关,假设 $f_i.a$ 表达动作,$f_i.o$ 表达客体,u_1“意图”中的功能描述与 u_2“收益”中的功能描述的功能相关度 $sim(f_1,cf_2)$使用如下公式计算:

$$sim(f_1,cf_2)=\frac{sim(cf_2.a,f_1.a)+sim(cf_2.o,f_1.o)}{2} \tag{2}$$

执行依赖关系关联强度 $S_{de}(u_1,u_2)$等于共同功能特征属性比例与功能相关度的加权平均,计算公式如下:

$$S_{de}(u_1,u_2)=\beta_1\frac{|V_1\cap V_2|}{|V_1\cup V_2|}+\beta_2\max(sim(f_1,cf_2),sim(f_2,cf_1)) \tag{3}$$

其中,$\beta_1+\beta_2=1$,$S_{de}(u_1,u_2)$是一个 0-1 之间的小数.

3) 近义关系关联强度计算

近义关系关联强度从 4 个维度度量近似关联强度:故事名、角色、功能特征属性集以及意图(动作、客体).假设 t_1,t_2 分别表示两个故事的标题,m_1,m_2 分别表示两个故事的意图,计算意图的相似性参见公式(2).

近义关系关联强度 $S_{si}(u_1,u_2)$等于上述 4 部分概念的加权平均,计算公式如下:

$$S_{si}(u_1,u_2)=\gamma_1sim(t_1,t_2)+\gamma_2sim(r_1,r_2)+\gamma_3\frac{|V_1\cap V_2|}{|V_1\cup V_2|}+\gamma_4sim(m_1,m_2) \tag{4}$$

其中,$\gamma_1+\gamma_2+\gamma_3+\gamma_4=1$,$S_{si}(u_1,u_2)$是一个 0-1 之间的小数.

4.2 用户故事冲突检查

冲突是指用户故事之间存在语义冗余或不一致,语义冗余是指两个用户故事表达了基本相同的语义,需要从故事描述、业务功能属性描述以及场景(状态变迁图)这 3 个方面分别检查.不一致是指两个用户故事的场景描述的状态变迁存在逻辑问题,如表 5 中的情况 4~情况 6.

对于两个用户故事 $u_1=(U_1,V_1,G_1)$和 $u_2=(U_2,V_2,G_2)$,其中,U 表示故事描述,V 表示功能特征属性集,G 表示状态变迁图.冲突检查需要分别检查这 3 个部分的匹配情况.具体过程如下.

1) 当角色相同且意图相近时,认为两个用户故事的故事描述匹配.角色和意图是否相近的判别方法见第 4.1 节.

2) 如果两个用户故事的功能特征属性集相同 $V_1=V_2$,则两个用户故事的功能特征属性集匹配.

3) 匹配状态变迁图的节点,通过检查合并规则发现冲突.将 G_1 中的每个状态与 G_2 中的状态进行匹配,如果发现了匹配的状态,再查看该状态相关联的状态变迁是否匹配,然后应用表 5 中的合并规则,检查是否有冲突发生.

冲突检查算法如算法 2 所示.

算法 2. 用户故事间的冲突检查.

输入:两个近义关系的故事功能场景表示 $fsNode_1$, $fsNode_2$;

输出:冲突检查结果.

伪代码:

```
1. U1 = getstorydesp(fsNode1),U2 = getstorydesp(fsNode2) //从两个功能场景表示中分别提取故事描述
2. V1 = getAttribute(fsNode1),V2 = getAttribute(fsNode2) //从两个功能场景表示中分别提取功能特征属性
3. G1 = getScenarioGraph(fsNode1),G2 = getScenarioGraph(fsNode2) //从两个功能场景表示中分别提取状态
                                                                  变迁图
4. if (match(U1,U2) = true)
5.       if (match(V1,V2) = true)
6.             for each state  S1 ∈ G1  do
7.                   for each state  S2 ∈ G2  do
```

8. if($match(S_1,S_2)$ = true)
9. if($findMatchofStateChage(S_1,S_2)$ = true) //找到匹配的状态变迁
10. print: 有冗余场景
11. else
12. if($matchConflictRule(merge(S_1,S_2))$ = true) //如果匹配相同状态后,违反了一致性
13. print: 有不一致场景
14. if($nomatch(S_1,T_2)$ = true) //S_1 在 T_2 中没有发现匹配,且与它相连的状态也无匹配
15. for each state $S_i \in T_1$ and S_i transition S_1 and $nomatch(S_i,T_2)$ do
16. print: 没有冲突发生

4.3 系统功能场景生成与更新

系统功能场景生成以一组功能特征表示为输入,计算机自动推荐关系,用户通过关系确认,最终生成系统功能场景.生成过程主要步骤如下.

• Step 1:输入一组带场景的用户故事,执行用户故事场景抽取,生成一个功能特征节点集合.

• Step 2:使用用户故事关系挖掘方法计算故事之间的合作关系、执行依赖关系以及近义关系的关联强度,根据关联强度的值,将大于阈值(根据领域问题和用户故事数据集的规模设定)的候选用户故事关系反馈给领域用户(关联强度值越高,相关度越高).

• Step 3:对于存在近义关系的故事,检查是否存在冲突.如果有冲突发生,报告给开发者.

• Step 4:领域用户根据用户故事关系挖掘的候选关系结果确认用户故事之间的关系.

• Step 5:按照确认的用户故事关系,在功能特征节点间建立关联边,生成系统功能场景.

系统功能场景更新也是一个交互的过程.领域用户提出删除、修改和添加新用户故事的申请.对于删除和修改的用户故事,自动化的方法是首先检索关联故事,反馈给领域用户确定删除和修改的故事.如果领域用户确认删除故事,则自动化地删除故事及其关联的边.对于修改后的用户故事或新加入的用户故事,自动化方法首先将用户故事结构化,然后识别依赖关系,进行冲突检查后反馈可能添加的位置信息.领域用户最终确认添加位置,将新增的用户故事加入到系统功能场景中.

5 系统功能场景交互式构建案例研究

本节结合在线购物系统案例,讨论系统功能场景构建及冲突检测的可行性.包括案例介绍、实验结果展示以及实验结果分析与讨论这 3 个部分.

5.1 案例介绍

我们以“在线购物系统”的带场景的用户故事集为案例,开展实验研究.首先由 3 名同学组成了一个研讨小组.分别从客户、商业用户和快递员 3 类用户的角色,使用本文中描述的带场景的用户故事的模板和表 2 中的推荐句型,撰写和提交需求文档.这 3 名同学共提交了 24 个用户故事,共计 57 个场景.业务功能操作客体(用户故事功能操作的对象)包括账号、商品、订单、联系人信息以及购物车等.表 6 为收集到的初始版本用户故事相关信息统计表.

Table 6 Characteristics of a set of user stories for a case study

表 6 案例用户故事集相关信息

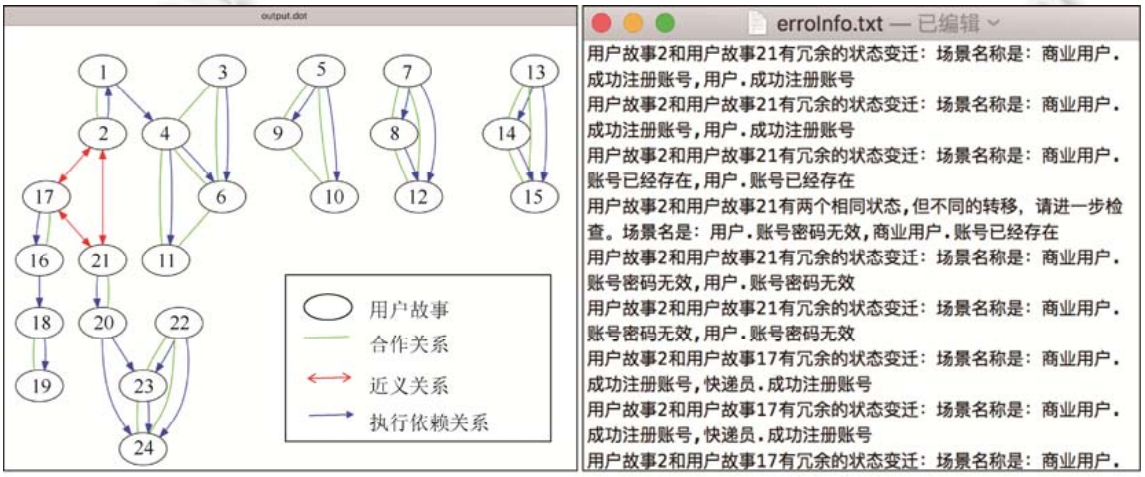
角色	用户故事数	场景数	功能操作客体及描述该客体的用户故事数
客户	15	32	账号(2),联系人信息(3),购物车(3),订单(3),商品(4)
商业用户	5	13	账号(2),商品(3)
快递员	4	12	账号(2),订单(2)

5.2 实验结果展示

为了验证本文提出的系统功能场景生成方法和表达用户故事之间关系的可用性,使用 Java 语言开发原型系统.该系统交互式地生成系统功能场景,包括 3 个步骤:首先,系统推荐 3 种用户故事关系(合作关系、执行依赖关系以及近义关系),然后,领域用户从推荐的关系中选择他们认为正确的关系,最后,系统根据用户确认的关系结果生成系统功能场景图并反馈给领域用户.同时,系统检测并反馈近义关系中可能存在的冲突.图 8 所示为原型系统推荐用户故事关系和领域用户确认关系过程的截图,图 9 所示为原型系统生成的系统功能场景图及存在近义关系的用户故事冲突反馈结果截图.

Fig.8 Screenshot of user-interface about recommendation and confirmation of user stories' relationships

图 8 用户故事关系推荐与确认交互界面截图



系统功能场景展示　　近义关系冲突检测结果

Fig.9 Screenshot of system function-scenario graph and the results of conflict defections

图 9 系统功能场景及冲突检查结果界面截图

图 9 所示的"系统功能场景展示"表达了 3 种用户故事关系,其中,合作关系(由无向边连接)将 24 个用户故事连接成 9 个连通区域,分别对应表 6 中角色执行业务客体的功能需求.每个连通区域中的用户故事共同合作实现具体"角色"针对一个具体"业务实体"(功能操作客体)的功能需要.例如,"1 和 2"("注册账号"和"登录账号")

表示用户对账号的功能需要.执行依赖关系(由单向有向边连接)将 24 个用户故事连接成 6 个连通区域,分别对应于账号、商品、订单、联系人信息以及购物车等业务实体的用户故事(功能)执行依赖关系,表达客户对购买商品、快递员配送商品、客户添加联系人信息、客户修改购物车、客户管理订单以及商业用户管理商品这 6 个功能执行次序.近义依赖关系(由双向有向边连接)表示用户故事有可能存在语义冗余,可以考虑对这些语义冗余进行合并.例如,案例中分别对客户、商业用户以及快递员描述了注册账号的用户故事,它们之间有着较高的冗余语义,可以考虑将 3 个故事合并.

5.3 实验分析与准确性影响

为了检验用户故事之间关系推荐的准确率和召回率,以及 3 种关系推荐时不同的反馈阈值(实验参数)的影响,进行了实验分析.准确率表明推荐关系的准确性,等于用户确认的数量除上系统推荐的数量.召回率表明推荐关系的完整性,等于用户确认的数量(系统推荐正确的数量)除上用户确认的数量与用户添加的关系数量的和.推荐关系反馈阈值表示当两个用户故事之间的关联程度大于反馈阈值时,该关系显示在推荐列表中.

Table 7 Statistic table of precision and recall about user stories' relationships that are recommended by system

表 7 用户故事关系推荐结果准确率与召回率统计表

关系名称	合作关系		执行依赖关系		近义关系	
推荐阈值	0.8	0.9	0.7	0.8	0.8	0.9
推荐关系数	22	14	33	14	6	3
确认关系数	21	14	18	9	3	3
添加的关系数	0	7	3	12	0	0
准确率(%)	95.45	100	54.5	64.28	50	100
召回率(%)	100	66.67	87.5	42.86	100	100

表 7 表示 3 种关系在不同推荐阈值时准确率和召回率的统计结果,其中,合作关系及近义关系有较好的准确率和召回率.执行依赖关系相对较差,原因是每个用户故事及其场景在表达上一般相对独立,很难从有限的信息中挖掘执行先后关系.结合领域知识可以得到更好的效果[24−26],将在后续工作中进行尝试.

推荐关系和冲突反馈结果的准确率还受到以下几个方面的影响.

(1) 用户故事描述语句格式的规范性:规范的语句格式便于应用标准模板提取用户故事概念语义,形成正确、完整的功能特征属性集以及状态变迁图,有助于准确地建立关联.

(2) 多人合作时领域术语的统一:领域术语统一对计算关系语句同义性起着重要作用.如两个需求提供者分别用“账号”和“帐号”表达同样的指代,但是它们的相似度并不是 1.

(3) In order to 字段中执行依赖语义的表达:本文描述的执行依赖关系的推荐依据主要来自于用户故事中 In order to 字段的描述信息.如果领域用户在写需求时省略了此项,或者没有表达相关语义,本文的方法不能准确地发现它们之间隐藏的执行依赖关系.只能通过人工添加的方法加入到系统功能场景.

(4) 自然语言处理方法的影响:本文使用开源工具 HanLP 进行自动化的自然语言分析,包括分词及依存句法分析.分析结果对关系的推荐结果有一定的影响.另外,计算词语之间的相似度是推荐挖掘的基础.但目前对词语中语义(特别是面向领域问题)的相似性计算效果并不理想.原型系统结合基于词的语义距离、NGram 和 word2Vec 的方法计算词语之间的相似度,得到较好的效果.

6 相关工作

敏捷开发方法由于具有快速应对需求变更的能力,被广大软件产品开发团队使用.团队成员之间的需求理解以及不同迭代版本的需求管理是有效发挥敏捷开发优势的关键.为解决这些关键问题,近年来,研究者们在用户故事与建模以及需求变更管理方面开展了一系列研究工作.

6.1.1 用户故事与建模

在 80%的敏捷方法中,使用用户故事表达客户对系统的需求[16,27,28].好的用户故事应相互独立,涉及小的功能,且需求可评估可测试[1−3,15,16].多个用户故事之间存在关联,它们组合在一起,形成完整的系统需求.这需要开

发人员能从系统的角度识别出一组用户故事之间的关系.很多工作主要提供可视化工具,以便于开发人员浏览和编辑需求[11,12,29].

Wautelet 等人[12]提出建立 Use-case 与用户故事之间的映射关系,该方法将用户故事中的角色映射到 use case 中的 actor,从意图和收益中分解任务,抽取任务及目标作为 Use-case 中的节点,并在目标与任务之间识别包含关系与扩展关系映射到 use case 中.他们提供了图形编辑工具[29],支持开发人员从用户故事、Use-case 以及类图 3 个角度观察需求.Mesquita 等人[11]提供了图形编辑工具 US2StarTool,支持将一组用户故事映射为一个 i^*模型,帮助开发人员以系统的视角考察参与者如何达成目标以及目标之间的关系.Trkman 等人[10]提出从用户故事中提取本体以构建业务过程模型(business process model),建立用户故事之间的顺序依赖关系.Wautelet[30]使用面向目标的方法建立多个用户故事之间的关系.Lucassen 等人[9,31]提出,借助自然语言分析的方法从用户故事中提取概念模型.他们总结出提取概念和关系的启发式规则,并根据规则抽取概念并建立关联关系.

这些方法都提供了建立用户故事与系统模型间的映射关系的手段,但是由于用户故事基本上是非结构化的,而且用户故事描述的信息过于简化,目前的方法大多通过可视化并结合人工建立用户故事关系.本文提出的方法借助带场景的用户故事模型抽取与关系挖掘,以人机协同的方式半自动化地建立用户故事之间的关联.最终形成一个系统的视角,帮助开发者了解用户故事之间的关系以及支持迭代的模型增加与修改.

6.1.2 需求变更管理

需求演进由需求变化引起[24,32],需求变化包括增加需求、删除需求或者修改需求.当需求规模较大时,分析和评估跨版本的需求变更以及随之带来的管理成本和风险变得更加困难.

一些研究工作着眼于收集需求数据,然后分析需求变化,进而发现新需求.Galvis 等人[33]从一组相关软件的需求反馈中自动提取主题,建立主题模型,用于分析变化的需求以及创新的需求.Schneider 等人[34]提出识别软件版本演化过程中用户对软件使用的反馈或建议,通过分类和整理从中获取新的需求.

为管理需求变化带来的风险,Shi 等人[18]从具有演化特征的多个连续版本的需求数据集中学习变化规律,以预测后续版本中需求可能发生的变化.Anderson 等人[19]以航空电子工业案例为例开展实证研究,通过对进化趋势和需求成熟度度量进化信息,以便理解需求演进.

在敏捷开发过程中,利益相关者之间的频繁交互导致迭代的需求过程,使得需求随着时间的推移变得更加清晰,有利于加强与客户的关系,并希望能以较少的时间投入,得到明确的系统需求[35].需求变更管理是其中的一个挑战性问题.目前需求变更管理大部分是通过维护产品的功能列表和索引卡来跟踪需求变化[2,36]的,对于一个规模较大的系统,要耗费大量的人力.Ajmeri[20]提出使用需求分析平台,支持多用户间的协同,促成敏捷需求的识别、讨论与定义.该方法提出使用领域知识语料(包括核心用户故事,行业规则,use case,数据模型,原型等),帮助客户、领域专家和其他分析人员能够在平台上修改和增加领域知识以适应具体项目的需要,提高敏捷开发中需求变更的管理能力.目前,也有研究从群体用户中采集需求,借助群体的智慧增进需求的理解,逐步提出更明确的需求,以支持需求的持续演化[37−39],如 Ali 等人[37]召集一组学生扮演不同视角的用户群体,从用户故事、场景到 use case 逐步提供明确的需求.最终得到一个完整的系统需求.

在需求变更管理中,每次需求变更都是一次繁琐的过程,涉及识别需求变更、变更分析及变更成本分析等步骤.我们的方法以系统模型为基础,提供一种人机协同的方式,支持需求变更.该方法同时考虑了需求变化可能会产生的冲突,根据冲突检查规则,将发现的冲突反馈给使用者.最终由使用者决策如何消解这些冲突.

7 结语和展望

在敏捷开发过程中,需求并不是事先定义好的,而是通过利益相关者之间的频繁交互,随着迭代开发过程逐步形成的.开发者理解领域用户提交的需求仍然是敏捷开发的瓶颈之一.一方面,用户故事所表达的信息有限,开发者要从功能实现的角度理解用户故事,需要花费很长的时间与领域用户讨论确认.另一方面,多个用户故事孤立地表达了领域用户的期望,缺乏系统的视角,开发者要理解用户故事之间的关系,推断其优先级也存在难度,导致系统模型构建上的困难.此外,当用户修改或提交新的用户故事时,确定新增用户故事在当前系统中的

位置,以及检查其与原有用户故事的冲突,都是繁重的工作.

为了减轻开发者的负担,发挥领域用户在特定领域上的专家优势,从而实现高质、高效的系统需求管理,本文提出并设计了一种人机协同的系统故事模型创建及更新方法.面对大量用户故事数据,让计算机自动地进行需求抽取和关联关系挖掘.领域用户对挖掘结果进行确认,人机合作完成系统功能场景图的构建和更新.案例研究表明,本文提出的用户故事整合方法具有较好的效果,具备一定的可行性.

需求提取与系统需求整合的质量与用户故事表达的质量相关.自然语言描述固有的模糊性,领域用户也不了解如何从独立的用户故事中整合出系统的功能需求,是影响需求质量的主要原因.从系统视角进行需求质量的检查与评估,以及提供一个支持敏捷团队需求协商的可视化平台,是后续工作中要解决的问题.

此外,在需求演化过程中,需求变更可能会对原来版本的需求产生冲突,我们目前的做法是根据冲突发现规则生成冲突报告.根据用户故事之间的关系的性质和领域知识可以给冲突消解提供参考,将更有效地指导需求工程师解决冲突问题,这是另一个后续工作中要解决的问题.

References:

[1] Boehm B, Turner R. Management challenges to implementing agile processes in traditional development organizations. IEEE Software, 2005,22(5):30–39.

[2] Cao L, Ramesh B. Agile requirements engineering practices: An empirical study. IEEE Software, 2008,25(1).

[3] Cohn M. User Stories Applied: For Agile Software Development. Addison-Wesley Professional, 2004.

[4] 王海峰.人工智能不会取代程序员,而是让编程更高质高效.见:第 22 届中国国际软件博览会高峰论坛.2016.

[5] Solis C, Wang X. A study of the characteristics of behaviour driven development. In: Proc. of the 37th EUROMICRO Conf. on Software Engineering and Advanced Applications (SEAA). IEEE, 2011. 383–387.

[6] Wynne M, Hellesoy A, Tooke S. The Cucumber Book: Behaviour-driven Development for Testers and Developers. 2nd ed., Pragmatic Bookshelf, 2017.

[7] Ambler S. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. John Wiley & Sons, 2002.

[8] Wautelet Y, Heng S, Kolp M, *et al*. Building a rationale diagram for evaluating user story sets. In: Proc. of the 10th IEEE Int'l Conf. on Research Challenges in Information Science (RCIS). 2016. IEEE, 2016. 1–12.

[9] Lucassen G, Robeer M, Dalpiaz F, *et al*. Extracting conceptual models from user stories with Visual Narrator. Requirements Engineering, 2017,22(3):339–358.

[10] Trkman M, Mendling J, Krisper M. Using business process models to better understand the dependencies among user stories. Information and Software Technology, 2016,71:58–76.

[11] Mesquita R, Jaqueira A, Agra C, Lucena M, Alencar F. US2StarTool: Generating i* models from user stories. In: Proc. of the Int'l i* Workshop (iStar). 2015. 103–108.

[12] Wautelet Y, Heng S, Hintea D, Kolp M, Poelmans S. Bridging user story sets with the use case model. In: Link S, Trujillo JC, eds. Proc. of the ER Workshops. 2016. 127–138.

[13] Kof L. Natural language processing for requirements engineering: Applicability to large requirements documents. 2004. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=2A6C7AF3FAC6D88BD969051D97D60B9F?doi=10.1.1.61.5849&rep=rep1&type=pdf

[14] Robeer M, Lucassen G, van der Werf JMEM, *et al*. Automated extraction of conceptual models from user stories via NLP. In: Proc. of the 24th IEEE Int'l Conf. on Requirements Engineering (RE). IEEE, 2016. 196–205.

[15] Cockburn A. Agile Software Development. Boston: Addison-Wesley, 2002.

[16] Leffingwell D. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional, 2010.

[17] Zowghi D, Gervasi V. On the interplay between consistency, completeness, and correctness in requirements evolution. Information and Software Technology, 2003,45(14):993–1009.

[18] Shi L, Wang Q, Li M. Learning from evolution history to predict future requirement changes. In: Proc. of the 21st IEEE Int'l Requirements Engineering Conf. (RE). IEEE, 2013. 135–144.

[19] Anderson S, Felici M. Quantitative aspects of requirements evolution. In: Proc. of the 26th Annual Int'l Computer Software and Applications Conf., COMPSAC. IEEE, 2002. 27–32.

[20] Ajmeri N, Sejpal R, Ghaisas S. A semantic and collaborative platform for agile requirements evolution. In: Proc. of the 3rd Int'l Workshop on Managing Requirements Knowledge (MARK). IEEE, 2010. 32–40.

[21] http://hanlp.linrunsoft.com/

[22] Goldberg Y, Levy O. Word2vec explained: Deriving Mikolov et al's negative-sampling word-embedding method. arXiv Preprint arXiv: 1402.3722, 2014.

[23] https://www.ltp-cloud.com

[24] Jin Z, Liu L, Jin Y. Software Requirement Engineering: Principles and Methods. Beijing: Science Press, 2008 (in Chinese).

[25] Rosadini B, Ferrari A, Gori G, *et al*. Using NLP to detect requirements defects: An industrial experience in the railway domain. In: Proc. of the Int'l Working Conf. on Requirements Engineering: Foundation for Software Quality. Cham: Springer-Verlag, 2017. 344–360.

[26] Kaiya H, Saeki M. Using domain ontology as domain knowledge for requirements elicitation. In: Proc. of the 14th IEEE Int'l Conf. on Requirements Engineering. IEEE, 2006. 189–198.

[27] Wang X, Zhao L, Wang Y, Sun J. The role of requirements engineering practices in agile development: An empirical study. In: Proc. of the Asia Pacific Requirements Engineering Symp. (APRES). 2014,(432):195–209.

[28] Kassab M. The changing landscape of requirements engineering practices over the past decade. In: Proc. of the Int'l Workshop on Empirical Requirements Engineering (EmpiRE). IEEE, 2015. 1–8.

[29] http://www.isys.ucl.ac.be/descartes/index.php

[30] Wautelet Y, VelghenM, Heng S, Poelmans S, Kolp M. On modelers ability to build a visual diagram from a user story set: A goal-oriented approach. In: Proc. of the REFSQ. 2018. 209–226.

[31] Lucassen G, Dalpiaz F, van der Werf JMEM, Brinkkemper S. Improving agile requirements: The quality user story framework and tool. Requirements Engineering, 2016,21(3):383–403.

[32] Anderson S, Felici M. Requirements evolution from process to product oriented management. In: Proc. of the Int'l Conf. on Product Focused Software Process Improvement. Berlin, Heidelberg: Springer-Verlag, 2001. 27–41.

[33] Galvis Carreño LV, Winbladh K. Analysis of user comments: An approach for software requirements evolution. In: Proc. of the 2013 Int'l Conf. on Software Engineering. IEEE Press, 2013. 582–591.

[34] Schneider K. Focusing spontaneous feedback to support system evolution. In: Proc. of the 19th IEEE Int'l Conf. on Requirements Engineering (RE). IEEE, 2011. 165–174.

[35] Inayat I, Salim SS, Marczak S, *et al*. A systematic literature review on agile requirements engineering practices and challenges. Computers in Human Behavior, 2015,51:915–929.

[36] Ramesh B, Cao L, Baskerville R. Agile requirements engineering practices and challenges: An empirical study. Information Systems Journal, 2010,20(5):449–480.

[37] Ali N, Lai R. A method of requirements elicitation and analysis or global software development. Journal of Software: Evolution and Process, 2017,29(4).

[38] Sharma R, Sureka A. CRUISE: A platform for crowdsourcing requirements elicitation and evolution. In: Proc. of the IC3. 2017. 1–7.

[39] Seyff N, Betz S, Groher I, Stade MJC, Chitchyan R, Duboc L, Penzenstadler B, Venters CC, Becker C. Crowd-focused semi-automated requirements engineering for evolution towards sustainability. In: Proc. of the RE. 2018. 370–375.

附中文参考文献:

[4] 王海峰.人工智能不会取代程序员,而是让编程更高质高效.见:第 22 届中国国际软件博览会高峰论坛.2016.

[24] 金芝,刘璘,金英.软件需求工程:原理和方法.北京:科学出版社,2008.

王春晖(1979－),女,内蒙古通辽人,讲师,CCF 专业会员,主要研究领域为软件工程,需求工程.

金芝(1962－),女,博士,教授,博士生导师,CCF 会士,主要研究领域为需求工程,知识工程.

赵海燕(1966－),女,博士,副教授,CCF 高级会员,主要研究领域为需求工程,软件复用,程序语言.

刘璘(1973－),女,博士,副研究员,主要研究领域为需求工程,信息系统工程,医疗数据分析.

张伟(1978－),男,博士,副教授,主要研究领域为群体化软件开发方法,软件需求工程.

崔牧原(1997－),男,学士,主要研究领域为软件工程,需求工程.