

面向问题的软件开发协同建模方法研究与实现

张晓¹ 李智^{1,2} 赵子岩¹ 付昌兰¹ 李伟东¹ 禹月昆¹ 王超¹

(广西师范大学计算机科学与信息工程学院 桂林 541004)¹

(广西师范大学广西多源信息挖掘与安全重点实验室 桂林 541004)²

摘要 建模软件是辅助需求工程师分析的工具,在需求设计阶段必不可少。目前,很少有需求建模工具可以集成能跨平台运行、支持在线多用户协同以及验证需求模型正确性和完整性等功能。鉴于问题框架方法在需求工程领域获得较大关注,本文开发了一款用户体验较好,兼容多平台的计算机辅助问题框架建模软件。本研究解决了自动化校验问题图的正确性和完整性以及复杂问题图拆分等难题,实现了用户登录,云端数据库存储设计和多人协同建模和验证,从而构建了一个在线需求建模、共享和验证的平台。

关键词 建模软件,问题框架,问题图拆分,协同建模

中图法分类号 TP311

文献标识码 A

DOI (投稿时不提供 DOI 号)

Research and Implementation of a Collaborated Modeling Approach for Problem-oriented Software Development

ZHANG Xiao¹ LI Zhi^{1,2} ZHAO Ziyang¹ FU Changlan¹ LI Weidong¹ YU Yuekun¹ WANG Chao¹

(College of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, China)¹

(Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin 541004, China)²

Abstract Software modeling tools are essential for assisting requirements engineers in system analysis during the requirements and design phase. At present, few existing requirements modeling tools can be run across different platforms, support online multi-user collaborations, and verify the correctness and completeness of requirements models. Since the Problem Frames (PF) approach is attracting much attention in the requirements engineering community, a computer-aided PF modeling tool, which provides good user experience and is compatible with multiple platforms, has been developed in this paper. The work in this paper has solved two difficult problems, i.e., automatic verification of the correctness and completeness of problem diagrams and mechanized decomposition of complex problem diagrams. Therefore, an online platform is built for requirements modeling, sharing and verification is established to support multi-user logins, deploy databases in the clouds and facilitate multi-user collaborations.

Keywords Modeling software, Problem Frames, Problem decomposition, Collaborated modeling

1 引言

需求分析作为软件开发过程中至关重要的一环,正日益受到人们的重视和关注。根据研究发现,软件项目中 40%-60% 的错误都是在需求分析和获取阶段产生的^[1]。美国权威组织

Standish Group 曾经对重大 IT 项目进行了研究调查,其中在 2001 年的报告显示,有多达 70% 的软件项目是以失败告终的,而导致这些项目失败的最主要原因是需求分析阶段的规格说明不完整。因此,需求分析的重要性不言而喻。于是软

到稿日期:2017-10-14 返修日期:

本文受广西研究生教育创新计划项目(XYCSZ2017066)基金资助。

张晓(1991-),男,硕士,主要研究方向为软件开发方法与理论, E-mail: 494471788@qq.com; 李智

(1969-),男,博士,教授,主要研究方向为软件需求工程、经验软件工程和软件测试, E-mail:

zhili@gxnu.edu.cn(通信作者)CCF 高级会员; 赵子岩(1991-),男,硕士,主要研究方向为软件需求工

程; 付昌兰(1990-),女,硕士,主要研究方向为软件需求工程; 李伟东(1989-),男,硕士,主要研究

方向为软件需求工程; 禹月昆(1991-),女,硕士,主要研究方向为软件需求工程; 王超(1993-),男,

硕士,主要研究方向为软件需求工程




件建模方法得到了广泛的重视,更有公司为统一建模语言(UML)和面向目标的 KAOS 方法开发了相应的辅助支持工具,提供给社会使用。


很多系统故障是由于对组件边界不明确的假设导致的。当做出确切的假设时,研发失败的原因很容易被识别和纠正。随着部署的软件的增加,系统变得越来越复杂和精细,增加了组件边界假设的难度。如果这些规约结合起来不建立在期望中的端到端需求上,即使所有的组件满足规约也是不够的^[2]。




问题框架方法提供了一种框架来描述软件与其它系统之间的交互。它帮助开发者理解软件问题所在的上下文,且它的很多方面与解决方案的设计相关。问题框架强调将端到端的软件需求分解为机器规约和一组领域属性。该方法由软件工程领域著名学者 Michael A. Jackson 最先提出^[3],后来经过不断的研究和发展,问题框架建模语言更加完善。相对于其他的软件开发方法,问题框架方法充分考虑周围环境对于软件系统的影响,对现实世界进行刻画,并且把需求的含义指标落实到现实世界相关领域的描述上。越来越多的研究者开始关注和研究问题框架,并取得了一定的进展,同时,问题框架的应用也越来越广泛。因此,国际上急需基于问题框架的建模工具。但是目前支持问题框架的建模工具非常少,因此完善和扩展问题框架辅助工具就变得很有必要。

传统的建模工具在协同和自动化检验图的完整性和正确性有些不足,例如:MS Visio, Rational Rose 和 UMLet 没有实现多用户之间共享文件和协同建模功能,对绘制出来的需求设计图不能够判断图的完整性和正确性。本文将着重介绍基于问题框架建模的计算机辅助支持工具,我们的研究在建模软件中的实现。

2 完整性、正确性验证研究

问题框架方法^[4]建模的模型定义了机器领域(machine domain, 表示)、问题领域(problem domain, 表示)和需求(requirement, 表示)。其中问题领域又进

一步分来设计领域(designed domain, 表示)和给定领域(given domain)。

机器领域与问题领域之间的连线称为接口交互(interface, 表示),需求与问题领域之间的连线称为需求引用(reference, 表示),需求与问题领域之间的连线称为需求约束(constraint, 表示)。

2.1 问题图的完整性、正确性条件

问题图的完整性是指在某些设计图中表现出来的信息不完善,领域结点和需求节点中缺少接口、约束和引用,缺少模型元素等造成不符合最基本的问题框架图规范。如表 1 所示。

表 1 问题图的完整性条件

编号	完整性条件
1	领域的名称必须被设置
2	上下文图中至少有一个机器领域
3	一个问题图/框架至少有一个需求
4	领域的缩写必须被设置
.....

问题图的正确性是指在设计图中模型元素之间需要满足问题框架方法的约束。如表 2 所示。

表 2 问题图的正确性条件

编号	完整性条件
1	领域的名称必须唯一
2	需求不能直接约束机器领域
3	每个机器领域至少控制一个接口
4	领域的缩写必须唯一
.....

当人们对需求进行建模的时候,希望软件能够帮助需求分析人员检查设计是否完整和正确,从而在一定程度上避免了因为设计不规范,而导致设计让开发人员难以理解。这里举例说明不完整和不正确的问题图,如图 1, 2 所示。图 1 中的问题领域没有名称,图 2 中需求不能和机器领域有约束。

当人们对需求进行建模的时候,希望软件能够帮助需求分析人员检查设计是否完整和正确,从而在一定程度上避免了因为设计不规范,而导致设计让开发人员难以理解。这里举例说明不完整和不正确的问题图,如图 1, 2 所示。图 1 中的问题领域没有名称,图 2 中需求不能和机器领域有约束。



图1 交通灯问题图（领域无名称）



图2 交通灯问题图（需求直接约束机器）

本文介绍的建模软件实现了问题图的完整性和正确性验证，为需求分析人员带来了极大的方便。1997 年 UML 采用对象约束语言 (OCL) 来定义约束。问题框架同 UML 一样，也可以使用 OCL 来定义。我们的研究借鉴 OCL 在 UML 模型一致性验证方法，我们用 OCL 定义问题框架约束，并通过分析 OCL 表达式来开发模型的完整性、正确性验证功能模块^[5]。同时因为实现验证的代码太长且不易懂，本文将使用 OCL 来描述问题框架模型约束。

2.2 OCL 简述

对象约束语言简称 OCL (Object Constraint Language)，它是一种用于施加在指定的模型元素上约束的语言。OCL 表达式以附加在模型元素上的条件和限制来表现对该对象的约束，其中包括附加在模型元素上的不变量或约束的表达式、附加在操作和方法上的前置条件和后置条件等^[6-7]。

2.3 OCL 描述完整性、正确性验证功能

首先申明 OCL 表达式中的变量 Class 和 Interface，分别表示包含所有的领域和接口的类。由于篇幅有限，这里用 OCL 仅描述表 1，2 中的问题图约束功能。

约束条件：领域的名称必须被设置

OCL 表达式：

```
Class.allInstances()->select(c|c.oclasType(
Class).getValue(c.oclasType(Class).getA
pppliedStereotypes()->asSequence()->firs-
t(),'value')=null) ->size()=0
```

约束条件：上下文图中至少有一个机器领域

OCL 表达式：

[键入文字]

```
Class.allInstances()->forall(p|p.oclasTy-
pe(Class).getAppliedStereotypes().name-
>includes('Machine')->size()>=1)
```

约束条件：一个问题图/框架至少有一个需求

OCL 表达式：

```
Class.allInstances()->forall(p|p.oclasTy-
pe(Class).getAppliedStereotypes().name-
>includes('Requirement') ->size()>=1)
```

约束条件：领域的缩写必须被设置

OCL 表达式：

```
Class.allInstances()->select(c|c.oclasTy-
pe(Class).getValue(c.oclasType(Class).getA
pppliedStereotypes()->asSequence()->firs-
t(),'abbreviation')=null) ->size()=0
```

约束条件：领域的名称必须唯一

OCL 表达式：

```
Class.allInstances()->select(getApplied-
Stereotypes().name->includes('Domain'))-
>isUnique(name)
```

约束条件：需求不能直接约束机器领域

OCL 表达式：

```
Interface.allInstances()->select(a|a.ocl-
AsType(Dependency).getAppliedStereotypes().
name->includes('constrains'))->forall(sou-
rce.getAppliedStereotypes().name->include-
s('Requirement')implies not target.getApp-
pliedStereotypes().name->includes('Machin-
e'))
```

约束条件：每个机器领域至少控制一个接口

OCL 表达式：

```
Interface.allInstances()->select(getApp-
pliedStereotypes().name->includes('obser-
ves')).target->forall(ot|Interface.all-
Instances()->select(getAppliedStereo-
types().name->includes('controls'))-
>select(target->exists(ct|ct=ot))->si-
ze()=1)
```

约束条件：领域的缩写必须唯一

OCL 表达式：

```
Class.allInstances()->select(c|c.oclAsType-
pe(Class).getValue(c.oclAsType(Class).getA
pppliedStereotypes()->asSequence()->firs-
t(),'abbreviation'))->isUnique(oclAsTy-
pe(String))
```

3 复杂问题拆分研究及实现

Hall^[8]为问题框架提供了一个指称语义可以用来表示整个问题图。问题图可以被表达如下：

$c, o : [K, R] = \{S \mid S \text{ controls } c \wedge S \text{ observes } o \wedge K, S \vdash_{DRDL} R\}$

图3是对应的通用问题图(注： $S!c$ 表示“ S controls c ”， $S?o$ 表示“ S observes o ”)：



图3 通用问题图

在本文中，引入因果链，以便对复杂问题图的理解，以及实现软件自动化查找因果链^[9]。

3.1 因果链

根据图3，可以看出如果领域共享现象 o 导致了 c 的发生，那么这是问题图中的因果关系。

在一个问题图中，如果找到一条由 $R \rightarrow S$ 或 $R \rightarrow S \rightarrow R$ 且含有 o 到 c 的路径。那么我们就说找到了问题的一个解，这条的路径上的元素构成了问题的解的集合。一个复杂问题图中会有多条这样的路径，也就是说有多条因果链。其中每条路径中的 R, S 可以分别表示为 $R1, S1, R2, S2, \dots, Rn, Sn$ 。

引入因果链概念后的Hall指称语义表达式如下：

$c, o : [K, R] = \{S \mid S!c \wedge S?o \wedge K, S \vdash_{DRDL} R\} = \{S1 \parallel S2 \parallel \dots \parallel Sn\}$

图4对应的是一个部分问题图，图中显示的是一条解的集合。

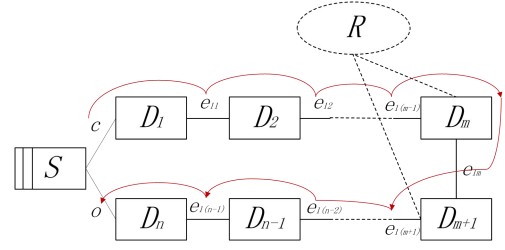


图4 部分问题图

3.2 电压监控器问题图案例分析

本文使用电压监控问题图案例分析，使读者更加理解因果链和Hall的指称语义表达式。

需要一台识别人脸特征的计算机来控制安全门，想要进入安全门的每个人的脸都被捕获到一个录像带上，用数据库中的项来与捕获的人脸特征进行比较，数据库中的项是已经被明确可以进入的人脸特征。如图5是安全门控制问题图。

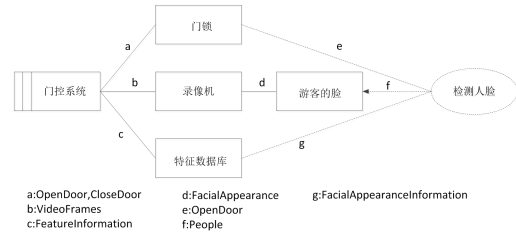


图5 安全门控制问题图

通过比较图3和图5，我们可以发现问题图中与Hall指称语义的关系以及2条因果链，图6所示。

$S = \{\text{门控系统}\},$

$c = \{a\}$ (b被S初始化，因此受S的控制)，

$o = \{b, c\}$ (b, c被S接收，因此受S的观察)，

$K = \{\text{门锁, 录像机, 游客的脸, 特征数据库}\},$

$R = \{\text{检测人脸}\},$

$d = \{e, f, g\}.$

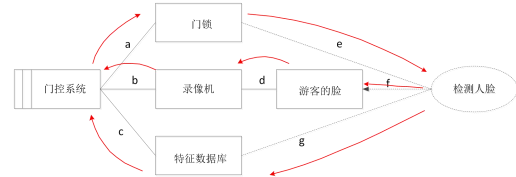


图6 安全门控制问题图因果链

4 多平台的建模软件设计及功能

本文中的问题框架建模软件采用的是客户端/服务器 (Client/Server, C/S) 和浏览器/服务器 (Browser/Server, B/S) 混合体系架构。因为两种体系架构各有优点, 所以混合体系架构能够节省开发和维护成本, 使系统具有良好的开放性、易扩展, 便于移植等特点^[10]。同时, 该建模软件集成了用户登录模块, 用户在离线或在线状态时都可以使用软件, 这便使得本建模软件在有无网络下都可以使用。有网络的情况下, 用户的数据可以永久保存在云端服务器数据库中。同时实现多人协同建模功能, 方便多个需求分析师不受地域限制和客户端软件的限制共同建模, 从而带来更好的体验。

4.1 C/S 和 B/S 混合的软件体系结构图

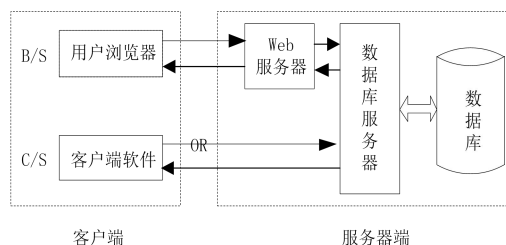


图7 问题框架协同建模软件体系结构图

4.2 软件协同建模功能的实现

因为多人协同建模, 难免会涉及设计上的分歧和同一个设计被其他用户篡改的冲突问题。因此本软件设计了用户模块, 每个用户都必须在软件官网注册帐号才可以使用在线建模功能, 没有帐号的用户仅可以在本地使用, 不能共享和保存设计。一个用户可以将自己的设计共享给其他人员参与设计, 其他人员的设计也会在数据库中同步。同时软件也会保留旧的设计, 从而达到版本控制的目的。如图8, 9, 10 界面截图。



图8 客户端软件用户登录界面



图9 浏览器访问用户登录界面

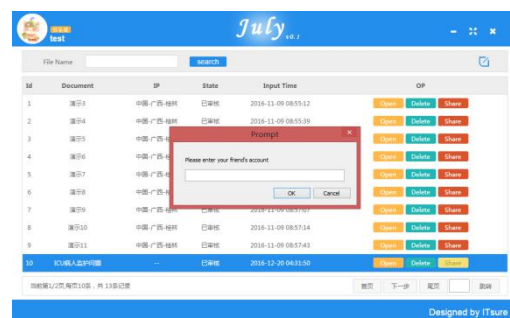


图10 共享设计图给其他用户

4.3 软件的主要功能

软件除了基本的绘制问题图的功能以外, 还实现了完整性、正确性校检问题图功能, 也能够根据因果关系找到因果关系链。绘制好的问题图是XML 格式, 用户可以上传自己的工程图到云服务器数据库中。需要修改的时候, 再从数据库中打开。我们先用软件绘制安全门控制问题图(图11 所示), 让软件自动校检图的正确性和完整性。

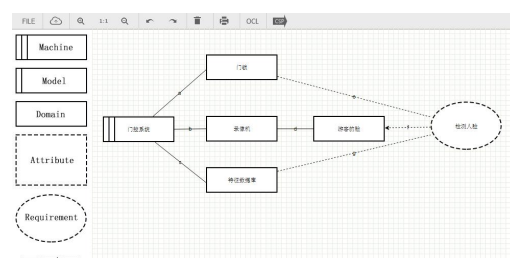


图11 绘制正确的问题图

[键入文字]

绘制好的问题图后，我们点击软件菜单中的 OCL 字样的按钮，即可以自动化校验问题图是否完整和正确(图 12 所示)。

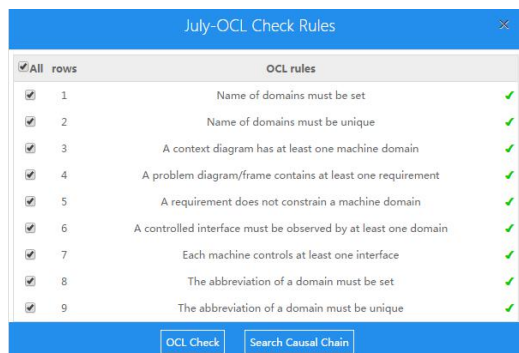


图 12 校验问题图结果截图

如果绘制不完整的或不正确的问题图，如图 13 所示，图中问题图的一个领域缺少名称。将绘制好的问题图进行完整性，正确校验结果如图 14 所示。

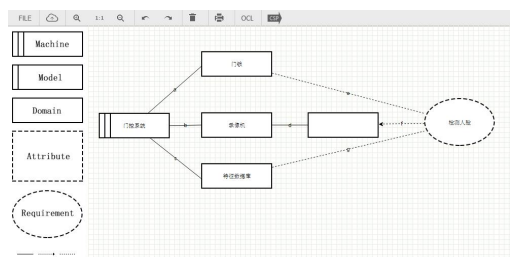


图 13 绘制错误问题图

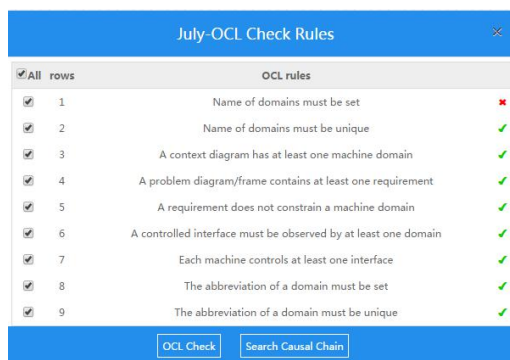


图 14 校验问题图结果截图

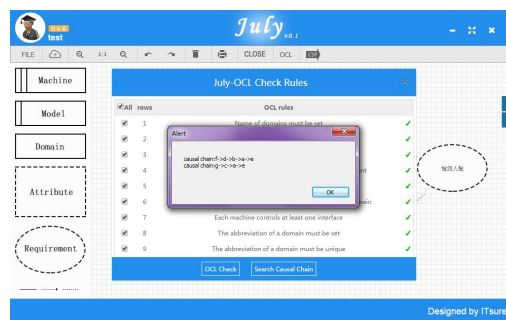


图 15 查找因果关系链结果截图

经软件检验后的正确问题图，可再用软件从问题图中找出所有的因果关系链(图 15 所示)。

在安全门控制案例中，共享现象 o 的集合中有 b 和 c 两个共享现象。共享现象 c 的集合有 a 一个共享现象。从而可以找到 2 条包含 o 到 c 的路径且是从需求到机器领域再到需求的因果关系链，即 $f \rightarrow d \rightarrow b \rightarrow a \rightarrow e$ 和 $g \rightarrow c \rightarrow a \rightarrow e$ 。

从软件给出的结果知，安全门控制需求模型中有 2 条因果关系链，与第 3 节案例中分析的结果一致。

结束语

在这篇文章中，我们提供了自动化校检问题图和查找复杂问题的子问题解即因果关系链。通过对传统建模工具的不足的提高需求设计的准确性，为尽可能的规避因需求阶段产生的错误而导致软件项目的失败，我们设计的软件可以实现多人协同完成同一个项目的需求建模。问题框架方法比较适合应用信息物理融合系统^[11]，而信息物理融合系统的设计是否建立在端到端的需求上，这需要我们引用因果关系链的成果，将问题图中的因果关系链转化成形式化脚本语言，进行需求满足性校验。因此，该软件可以更好的帮助需求工程师理解复杂问题和验证设计。

参考文献

- [1] Cleland-Huang J, Chang C K, Christensen M.Event-Based Traceability for Managing Evolutionary Change[J]. IEEE Transactions on Software Engineering, 2003, 29(9):796-810.

- [2] Seater R, Jackson D. Problem frame transformations: deriving specifications from requirements[C]// International Workshop on Advances and Applications of Problem Frames. ACM, 2006:71-80.
- [3] Jackson M. Problem frames: analyzing and structuring software development problems[M]. Addison-Wesley Longman Publishing Co. Inc. 2000.
- [4] 陈小红, 尹斌, 金芝. 基于问题框架的需求建模:一种本体制导的方法[J]. 软件学报, 2011, 22(2):177-194.
- [5] Queralt A, Teniente E. Verification and Validation of UML Conceptual Schemas with OCL Constraints[J]. Acm Transactions on Software Engineering & Methodology, 2012, 21(2):1-41.
- [6] Gogolla M. Object Constraint Language[J]. 2016.
- [7] 江泽凡, 王林章, 李宣东,等. 基于 UML 顺序图的测试方法[J]. 计算机科学, 2004, 31(7):131-136.
- [8] Li Z, Hall J G, Rapanotti L. On the systematic transformation of requirements to specifications[J]. Requirements Engineering, 2014, 19(4):397-419.
- [9] Jackson M. Where, Exactly, Is Software Development?[M]// Formal Methods at the Crossroads. From Panacea to Foundational Support. Springer Berlin Heidelberg, 2003:115-131.
- [10] Chen X, Liu J L. Analysis and Comparison between the Structures of Client/Server and Browser/Server[J]. Journal of Chongqing Institute of Technology Management, 2000.
- [11] M. Jackson. System Behaviours and Problem Frames: Concepts, Concerns and the Role of Formalisms in the Development of Cyber-physical Systems[M]. Dependable Software Systems Engineering, 2015:79-104.

请在文末添加作者的联系电话和邮箱！

+ 通讯作者: 李智

Phn: +86-0773-5811-621, Fax: +86-0773-5811-621,

E-mail: zhili@gxnu.edu.cn