计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(9):41−49 [doi: 10.15888/j.cnki.csa.007039]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

多视点元模型间需求追踪性方法①

李　潇, 魏长江

(青岛大学 数据科学与软件工程学院, 青岛 266071)
通讯作者: 李　潇, E-mail: 839779044@qq.com

摘　要: 以全面而统筹的眼光解决问题, 则必须从局部分析不同的侧面, 面向多视点的需求工程即为该理论的一个应用. 多视点需求工程越来越受到重视, 但是各视点间却缺少统一, 这不仅不利于获取完整的系统需求, 还导致需求变更困难等问题. 为了解决以上问题, 首先提出一个新的多视点建模过程框架, 合理的建模框架有利于获取完整的系统需求. 其次在多视点建模过程框架下建立追踪元模型, 阐明系统需求在多视点元模型间的平稳过渡. 最后结合需求追踪矩阵方法在追踪元模型间建立起追踪关系, 进而通过计算变化追踪矩阵在需求变更时可以追踪到相关元素并进行更改, 解决需求变更困难的问题.

关键词: 需求工程; 多视点; 追踪元模型; 需求追踪矩阵; 需求变更

引用格式: 李潇,魏长江.多视点元模型间需求追踪性方法.计算机系统应用,2019,28(9):41–49. http://www.c-s-a.org.cn/1003-3254/7039.html

Requirements Tracking Method among Multi-View Meta-Model

LI Xiao, WEI Chang-Jiang

(School of Data Science and Software Engineering, Qingdao University, Qingdao 266071, China)

Abstract: To solve a problem with a comprehensive and integrated vision, we must analyze different aspects from the perspective of locality and consider it in a divide-and-conquer manner. The multi-viewpoint requirement engineering is getting more and more attention, but there is a lack of unity among viewpoints. This not only fails to capture the complete system requirements, but also causes difficulties in changing requirements. In order to solve these problems, this study first builds a multi-viewpoint based modeling process framework, for a reasonable modeling framework is conductive to obtain complete system requirements. Secondly, the tracking meta-model under the new multi-viewpoint modeling process framework is set up, aiming to illustrate the smooth transition of system requirements across multi-viewpoint meta-model. Finally, a tracking relationship among the tracking meta-model with the requirement tracking matrix method is established, thus relevant elements can be tracked and changed by calculating the change tracking matrix as the requirement changes. The above research can solve the problem of difficulty in requirements change.

Key words: requirements engineering; multi-viewpoint; tracking meta-model; requirements tracking matrix; requirement change

统一建模语言 (Unified Modeling Language, 简称 UML) 是一种用于对软件系统的制品进行可视化、详述、构造和文档化的图形建模语言[1]. 传统的基于 UML 的面向对象的建模方法在不同视点下将建模过程划分为从概念到实现的若干个阶段[2], 包括用例模型、静态结构模型、动态行为模型和物理结构模型.

引入多视点的需求建模方法[3,4], 使得软件开发人员能够按照不同的视点将整体划分成分散、独立的若

① 收稿时间: 2019-02-27; 修改时间: 2019-03-15; 采用时间: 2019-03-18; csa 在线出版时间: 2019-09-05

干部分, 最大程度地捕获待开发系统及系统相关人员的完整需求信息. 此方法只关注视点内所感兴趣的问题, 降低了需求描述的复杂度, 并且各个视点之间相互补充和配合, 减少了需求的遗漏.

然而, 在不同的软件开发阶段和不同的涉众下建立的不同模型, 视点的分散性导致了模型间缺乏统一. 目前, 在传统的基于 UML 的面向对象建模方法中, 多视点下模型间关联困难的原因主要表现在三个方面[5]: (1) 四种视点下的 UML 模型缺乏对软件系统总体结构上的考虑; (2) 过早的专注软件系统的细节, 缺乏对软件系统整体的把握; (3) 对于软件系统的非功能需求的描述存在不足. 这不仅导致了无法获取完整的系统需求, 还难以进行需求变更管理.

目前, 针对需求变更困难问题的解决方法有很多种, 但被需求工作者广泛使用的是基于需求追踪的方法. 2013 年, Cimatti A[6]等人提出一种综合性方法来验证系统的功能型需求, 并且还实现了从非正式需求到正式、可追踪、自动化、可扩展需求的转换过程. 2014 年, Goknil A[7]等人分析并改进了需求中的变更影响, 对需求的可追踪性进行研究, 并开发 TRIC 工具对需求变更进行分类和验证. 2017 年, Tekinerdogan B[8]等人对需求的可追踪性提出了不同的要求, 在需求的元模型基础上分析并实现了需求的可追踪性. 除此之外, 国内关于获取一致性、易于变更且可追踪的需求问题也有相关的研究. 2005 年, 朱雪峰、金芝[9]提出了一种需求不一致性管理框架的需求建模方法, 就需求的不一致性管理方面有代表性的工作进行了较为系统的分析. 2006 年, 王映辉等[10]提出了一种用于处理需求变更问题的有效方法, 通过在关系矩阵中记录变化的系统需求, 然后通过矩阵的运算实现变更过程中需求的追踪.

因此, 本文一方面将面向目标的 I*模型和软件体系结构模型引入到传统的基于 UML (Unified Modeling Language) 的面向对象的建模阶段中, 旨在设计一个基于多视点的建模过程框架. 一方面在三个不同视点下建立追踪元模型, 阐明新的建模过程框架如何实现系统需求在多视点元模型间的平稳过渡. 另一方面通过计算需求追踪矩阵, 在多视点元模型间建立起可追踪关系, 实现对系统需求的追踪并验证.

1 元模型的建立

1.1 基于多视点的建模过程框架

在我们的研究领域中, 为了有效解决多视点下模型间关联困难的问题并获取完整的系统需求, 本文将建模过程划分为 I*[11]目标建模、用例建模以及软件体系结构[12]建模三个阶段, 旨在设计一个新的基于多视点的建模过程框架如图 1 所示. 在新建的建模过程框架中, 将面向目标的 I*模型建模在用例模型之前, 不仅能够避免过早的专注软件系统的细节, 而且能够有效描述系统的非功能需求; 将软件体系结构模型建模在用例模型之后, 通过 UML 构件图和部署图实现对软件系统的总体结构的描述.

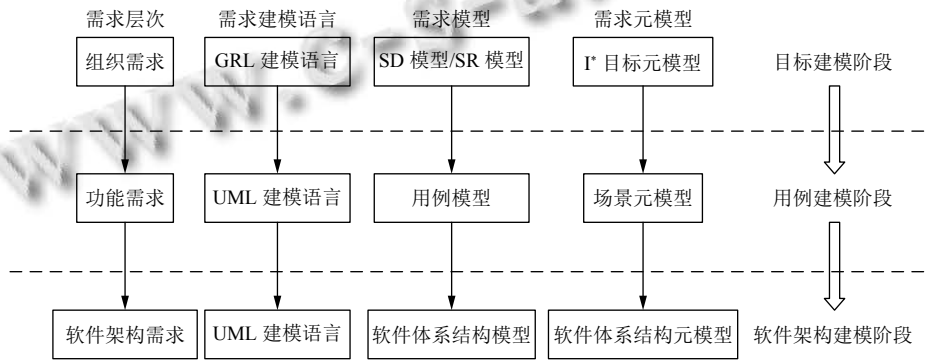

图 1 基于多视点的建模过程框架

1.2 建立 I*目标模型的元模型

1.2.1 I*目标建模技术简介

GRL (Goal-oriented Requirements Language)[13–15]作为一种可视化建模语言, 主要应用于面向目标建模和推理需求过程. 基于 I*框架的建模是一种面向目标的需求建模方法, I*模型中的元素由 GRL 语言提供,

并且支持对非功能需求的分析与建模.

I*目标建模框架是一种在早期阶段需求工程中着重描述软件环境中各参与者意图的建模方法[16]:该框架使用策略依赖模型 (Strategic Dependency model, 简称 SD 模型) 建立参与者意图间的依赖, 推导出能够使得意愿满足的策略; 另外使用策略原理模型 (Strategic Rational Model, 简称 SR 模型) 表示参与者的意图.

1.2.2 I*目标元模型的建模与分析

图 2 所示 I*目标模型的元模型元素关系分析如下: (1) AND: 表示元素间的 AND 分解关系. 他能够分解一个目标为一系列子目标, 表示父目标的实现依赖于全体子目标的共同实现; (2) OR: 表示元素间的 OR 分解关系, 它与 AND 分解关系类似, 不同的是父目标的实现可以由任意一个子目标 (子软目标) 的实现所代替; (3) 贡献: 定义为当目标被实现时, 该目标对软目标的影响关系; (4) 依赖: 不同参与者之间在某种元素关系上的依赖关系, 依赖的双方为依赖者 depender 和被依赖者 dependee.

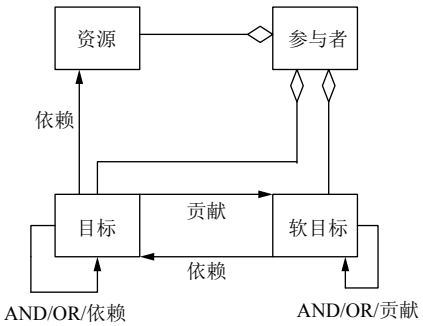

图 2 I*目标模型的元模型

1.3 建立场景元模型

1.3.1 用例建模技术简介

功能需求是指开发人员必须实现的软件功能或软件系统应具备的外部行为[17]. 目前, 对系统进行需求获取与分析最常用的技术是基于 UML 的用例技术, 虽然说用例不能对所有的系统需求进行可视化建模, 但却为功能需求提供了良好的支持并得到了广泛的应用[18].

用例着重于系统功能需求, 关注的是系统与外部参与者间的交互, 用于描述可发生的所有动作序列, 而场景则是特定的动作序列. 所以, 用例与场景的关系中, 场景是用例的一个实例, 用例则是对于主角与系统交互层面上的那些相关交互场景的集合的描述.

1.3.2 场景元模型的建模与分析

图 3 所示的场景元模型元素关系分析如下: (1) 场景根据满足目标的不同分为正常场景和异常场景; (2) 场景描述被描述为一个或多个动作的组合; (3) 动作分为动作流和原子动作, 动作流包括一组动作; (4) 场景会有状态的变化, 状态中的起始状态和终止状态分别标志着场景的开始与结束; (5) 代理执行相关的原子动作, 并对对象参数产生一定的影响.

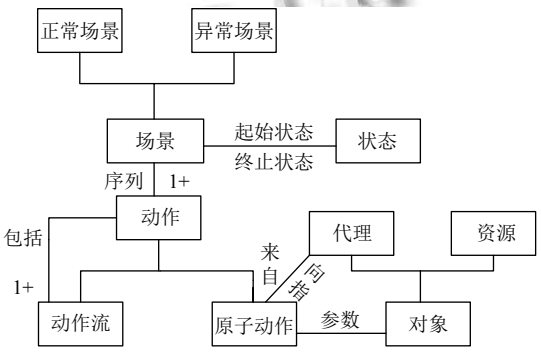

图 3 场景元模型

1.4 建立软件体系结构元模型

1.4.1 软件体系结构简介

软件体系结构[19,20]提供了一个抽象的系统规范, 主要包括描述系统功能的构件、构件之间的连接件、构件与外界环境交互的接口以及配置的约束等. 软件体系结构不仅给出了系统的总体结构, 而且显示了系统需求和构成系统的构件之间的对应关系.

1.4.2 软件体系结构元模型的建模与分析

如图 4 所示软件体系结构元模型元素关系分析如下: (1) 一组相互协作的构件 (components) 以及连接这些构件的连接器 (connectors) 定义了应用系统的软件体系结构; (2) 为了处理来自业务领域需求的不断变化以及软件技术的不断进步所引起的系统变更, 软件体系结构可以通过预先定义的一些扩展点来组装用户新开发的构件; (3) 从模型结构上看, 构件和连接器形成了软件体系结构, 此外还包含了配置 (configuration), 配置给出了系统的物理结构的文本形式; (4) 构件的接口由一组端口组成, 端口是构件与外部系统进行交互的纽带; (5) 连接件的接口由一组角色组成.

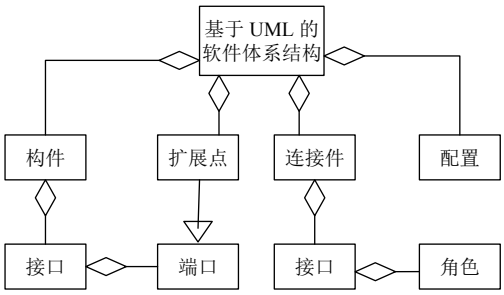

图 4 软件体系结构元模型

2 多视点元模型间的追踪

2.1 建立追踪元模型

本小节建立了追踪元模型如图 5 所示. 它由三个不同的层次组成, 包括目标层、场景层以及软件体系结构层. 对每个级别中的制品和追踪关系进行建模: 目标层涉及功能需求和非功能需求, 分别建模为“目标”和“软目标”; 场景层中涉及到为完成某个特定的目标而进行的一系列特定的“动作”; 因此, 我们将目标元模型与场景元模型间的追踪关系建模为“支持”, 表示一系列场景能够“支持”相应目标的实现; 软件体系结构元模型表示了系统需求和构成系统的“构件”间的对应关系, 软件构件技术是指通过软件的构件化把需要实现的功能分解成一系列标准的原子构件, 其中每个“构件”都具有特定的功能实现[21]; 因此, 我们将场景层与软件体系结构层的追踪关系建模为“实现”.

通过分析不同层次元模型元素含义以及它们间的关联关系, 实现了系统需求在 I*目标元模型、场景元模型、软件体系结构元模型间的平稳过渡.

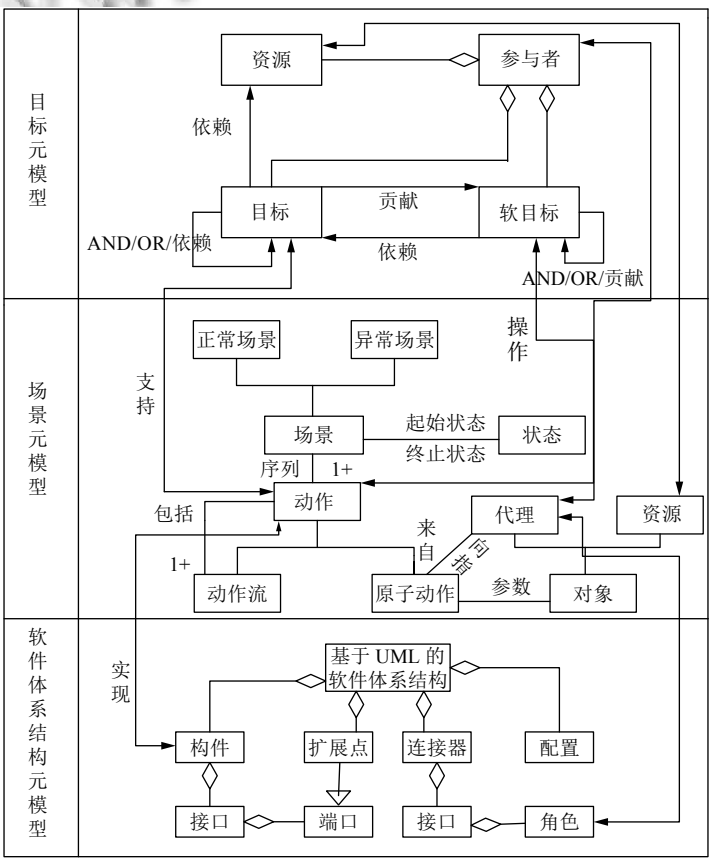

图 5 追踪元模型

2.2 需求传播途径

根据图 5 建立的追踪元模型可知, 在使用场景描述需求时, 根据需要完成的功能设置相关的目标, 为了实现这个目标就需要多个场景的支持, 而场景是由若干个动作组成的, 同时每个构件都具有特定的功能实现, 最后每个构件都被部署在不同的硬件节点上. 因此,

在追踪元模型中, 目标、场景、动作、构件中任何一个元素发生改变, 都可以根据目标 g (goal)→场景 s (scenario)→动作 a (action)→构件 c (component)→节点 n (node) 的传播途径进行追踪. 为了更清晰的表达追踪元模型各元素间关系, 得到如图 6 所示的需求传播途径.

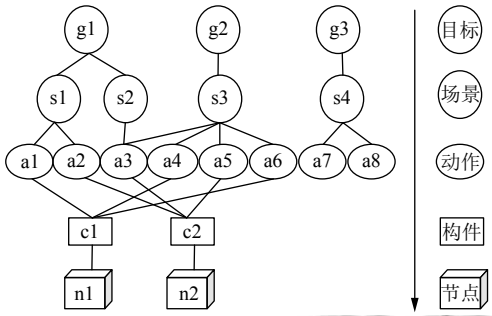

图 6 需求传播途径

2.3 需求追踪矩阵

需求追踪矩阵是一种主要管理需求变更和验证需求是否得到实现的有效工具, 可以跟踪每个需求的状态[22,23].

根据图 6 需求传播途径, 将元素间连线关系用具有一定语义的数值表示, 其中“1”标记元素间有关联, “0”表示元素间无关联, 可分别定义场景-目标追踪矩阵 Mg, 动作-场景追踪矩阵 Ms, 构件-动作追踪矩阵 Ma 以及节点-构件追踪矩阵 Mc.

建立以上四个追踪矩阵可有效管理需求变更, 当需求中的某个元素改变时, 可以根据追踪矩阵找到并更改这个元素所影响的其他元素. 通过 Mg 可以判断“目标”改变时会影响到的“场景”; 通过 Ms 可以判断“场景”改变时影响到的“动作”; 通过 Ma 判断“动作”改变时影响到的“构件”; 通过 Mc 判断“构件”改变时影响到的“节点”.

除了对系统需求在元模型间进行逐层的追踪, 还可以跨层追踪, 例如当“目标”发生变化时, 可通过计算矩阵 Mg–s 直接追踪到相关的“动作”, 其中

$$\begin{cases} Mg-\mathrm{s}=Ms\times Mg=m_{\mathrm{g-s}(i,j)},\ \ i,j=1,2,3,\cdots,n \\ m_{\mathrm{g-s}(i,j)}=\sum\limits_{x=1}^{k}\left(m_{\mathrm{s}(i,j)}\times m_{\mathrm{g}(x,j)}\right) \end{cases} \tag{1}$$

还可以通过计算矩阵 Mg-s-a 判断“目标”变化时影响哪些“构件”, 其中

$$\begin{cases} Mg-\mathrm{s}-\mathrm{a}=Ma\times(Ms\times Ma)=m_{\mathrm{g-s-a}(i,j)} \\ i,j=1,2,3,\cdots,n \\ m_{\mathrm{g-s-a}(i,j)}=\sum\limits_{x=1}^{k}\left(m_{\mathrm{a}(i,x)}\times m_{\mathrm{g-s}(x,j)}\right) \end{cases} \tag{2}$$

通过计算矩阵 Mg–s–a–c 判断“目标”变化时会影响哪些构件的部署, 其中

$$\begin{cases} Mg-\mathrm{s}-\mathrm{a}-\mathrm{c}=Mc\times Mg-\mathrm{s}-\mathrm{a} \\ \qquad\qquad =Mc\times(Ma\times(Ms\times Mg)) \\ \qquad\qquad =m_{\mathrm{g-s-a-c}(i,j)} \\ \qquad i,j=1,2,3,\cdots,n \\ m_{\mathrm{g-s-a-c}(i,j)}=\sum\limits_{x=1}^{k}\left(m_{\mathrm{c}(i,x)}\times m_{\mathrm{g-s-a}(x,j)}\right) \end{cases} \tag{3}$$

所以, 通过跨层追踪得到矩阵 Mg–s、Mg–s–a、Mg–s–a–c.

通过分析矩阵运算结果可追踪到相关的需求变化关系, 从而有效管理需求变更, 例如矩阵中第 i 行第 j 列中的非零数字即表示两个元素之间具有可追踪关系, 改变其中一个元素便会对另一个元素产生影响.

3 实例分析

本小节以自动取款机 (ATM) 系统为例, 对以上方法进行可行性分析与研究.

首先建立基于 ATM 系统的可追踪元模型如图 7 所示. 第一层表示功能需求和非功能需求的目标元模型级别; 第二层表示场景元模型的级别, 描述了为支持某个目标而进行的一系列活动; 第三层表示软件体系结构元模型级别, 描述了功能分配以及构件的部署.

在目标元模型层和场景元模型层间: “成功取款”、“查询账户信息”以及“基本操作”属于 ATM 系统的功能需求, 即目标. 一个目标的完成需要一系列的动作支持, 而动作的执行都是由目标所驱动, 例如“成功取款”目标由“取款”场景支持, 而此场景又包括“输入取款金额”、“验证取款金额”、“数据传输”以及“取出现金”这一系列动作. 又比如: “基本操作”目标由“发起会话”和“传输数据”场景支持, “插卡”和“验证密码”则是“发起会话”场景所包含的一系列动作. 对非功能需求而言, “ATM 设备安全”以及“ATM 系统安全”是从“ATM 安全”的目标中分解出来的非功能需求, 即软目标, 并且“ATM 系统安全”软目标可以通过“安装防病毒系统”和“及时升级测试”这两个动态操作来满足.

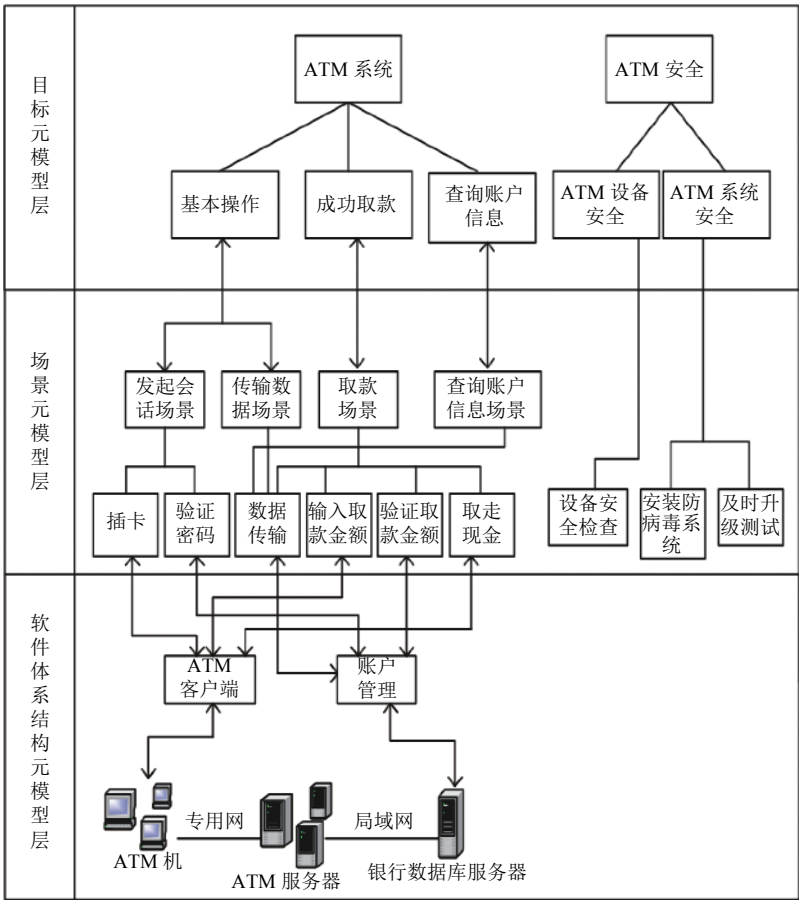

图 7 基于 ATM 系统的追踪元模型

在场景元模型层和软件体系结构元模型层间: 将 ATM 系统功能划分并分布到特定构件, 并且将特定功能的构件部署在节点上来展示系统的架构. 例如, “插卡”、“输入取款金额”以及“取走现金”的动作都在“ATM 客户端”界面上操作, 接着将“ATM 客户端”这一构件部署到“ATM 机”节点上. 同理, 将与“取款”功能相关的的“账户管理”构件部署在“银行数据库服务器”节点. 同时, “ATM 机”、“ATM 服务器”以及“银行数据库服务器”之间又通过特定的网络连接.

然后, 通过分析基于 ATM 系统部分功能的追踪元模型, 得到 ATM 系统部分需求的传播途径如图 8 所示.

接下来, 根据基于 ATM 系统的需求传播途径图, 得到场景-目标追踪矩阵 Mg, 动作-场景追踪矩阵 Ms, 构件-动作追踪矩阵 Ma 以及节点-构件追踪矩阵 Mc 为:

$$M\mathrm{g}=\begin{bmatrix}1&0&0\\1&0&0\\0&1&0\\0&0&1\end{bmatrix}\quad M\mathrm{a}=\begin{bmatrix}1&0&0&1&0&1&0&0\\0&1&1&0&1&0&0&0\end{bmatrix}$$

$$M\mathrm{s}=\begin{bmatrix}1&0&0&0\\1&0&0&0\\0&1&1&0\\0&0&1&0\\0&0&1&0\\0&0&1&0\\0&0&0&1\\0&0&0&1\end{bmatrix}\quad M\mathrm{c}=\begin{bmatrix}1&0\\0&1\end{bmatrix}$$

然后根据矩阵运算式 (1)、式 (2)、式 (3) 实现元模型的跨层追踪. 例如, 通过式 (1) 计算矩阵 Mg–s 建立 ATM 系统目标与动作间的追踪关系, 计算 Ms–a 建立 ATM 系统场景与构件间的追踪关系; 通过式 (2) 计算 Ma–c 建立 ATM 系统动作与构件间的追踪关系; 通过式 (3) 计算矩阵 Mg–s–a–c 实现 ATM 系统目标与节点的追踪, 最终实现不同视点下元模型的统一. 得到矩阵 Mg–s、Ms–a、Ma–c、Mg–s–a–c 分别如式 (4)–式 (7).

根据以上矩阵运算结果, 得到基于 ATM 系统的目标元模型与场景元模型间, 以及目标元模型与软件体系结构元模型间的追踪关系如图 9 和图 10 所示.

$$Mg-s=Ms\times Mg=\begin{bmatrix}1&0&0&0\\1&0&0&0\\0&1&1&0\\0&0&1&0\\0&0&1&0\\0&0&1&0\\0&0&0&1\\0&0&0&1\\0&0&0&1\end{bmatrix}\times\begin{bmatrix}1&0&0\\1&0&0\\0&1&0\\0&0&1\end{bmatrix}=\begin{bmatrix}1&0&0\\1&0&0\\1&1&0\\0&1&0\\0&1&0\\0&1&0\\0&0&1\\0&0&1\end{bmatrix} \tag{4}$$

$$Ms-a=Ma\times Ms=\begin{bmatrix}1&0&0&1&0&1&0&0\\0&1&1&0&1&0&0&0\end{bmatrix}\times\begin{bmatrix}1&0&0&0\\1&0&0&0\\0&1&1&0\\0&0&1&0\\0&0&1&0\\0&0&1&0\\0&0&0&1\\0&0&0&1\end{bmatrix}=\begin{bmatrix}1&0&2&0\\1&1&2&0\end{bmatrix} \tag{5}$$

$$Ma-c=Mc\times Ma\begin{bmatrix}1&0\\0&1\end{bmatrix}\times\begin{bmatrix}1&0&0&1&0&1&0&0\\0&1&1&0&1&0&0&0\end{bmatrix}=\begin{bmatrix}1&0&0&1&0&1&0&0\\0&1&1&0&1&0&0&0\end{bmatrix} \tag{6}$$

$$Mg-s-a-c=Mc\times Mg-s-a=\begin{bmatrix}1&0\\0&1\end{bmatrix}\times\begin{bmatrix}1&2&0\\2&2&0\end{bmatrix}=\begin{bmatrix}1&2&0\\2&2&0\end{bmatrix} \tag{7}$$

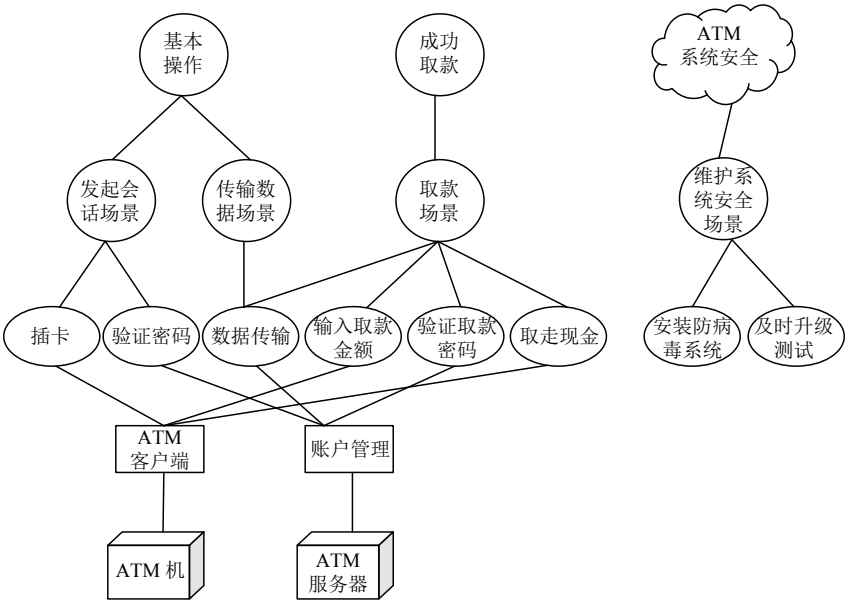

图 8 基于 ATM 系统的需求传播途径

以“成功取款”目标为例, 验证基于 ATM 系统的目标元模型层和场景元模型层间, 目标元素与动作元素追踪关系的准确性. 根据图 8 可知, “成功取款”目标更改影响到的动作是与“取款”场景相关的动作一致. 其中, “取款”场景由一系列“数据传输”、“输入取款金额”、“验证取款密码”、“取走现金”系统或人为的动作来完成. 这意味着 g2 与 a3、a4、a5、a6 间具有关联关系. 显然, 这与图 9 的第二列一致. 同理, 可知图 9 的其他列也是正确的.

	g1	g2	g3
s1	1	0	0
s2	1	0	0
s3	1	1	0
s4	0	1	0
s5	0	1	0
s6	0	1	0
s7	0	0	1
s8	0	0	0

图 9 ATM 系统中目标层与场景层间追踪关系

	g1	g2	g3
n1	1	2	0
n2	2	2	0

图 10 ATM 系统中目标层与软件体系结构层间追踪关系

再以“成功取款”目标为例, 验证基于 ATM 系统的目标元模型层和软件体系结构元模型层间, 目标元素和节点元素追踪关系的准确性. 根据图 8 可知, “成功取款”目标的更改影响到的构件是与“数据传输”、“输入取款金额”、“验证取款密码”以及“取走现金”动作相关的构件一致, 其中“数据传输”和“验证取款密码”与“账户管理”构件关联, “输入取款金额”和“取走现金”则与“ATM 客户端”构件关联, 而这两个构件又分别部署在“ATM 服务器”和“ATM 机”节点上. 这意味着 g2 与 n1、n2 间具有关联关系. 显然, 这与图 10 的第二列一致. 同理, 可知图 10 的其他列也是正确的.

因此, 通过矩阵运算, 在 ATM 系统多视点元模型间建立了可追踪关系, 一方面实现了在不同视点下元模型的统一, 另一方面可有效管理需求变更.

4 总结与展望

本文主要研究了一种在多视点的元模型间进行需求追踪的方法. 一方面把面向目标的 I*模型和软件体系结构模型引入到传统的基于 UML (Unified Modeling Language) 的面向对象的建模阶段中, 旨在设计一个基于多视点的建模过程框架. 合理的建模框架使我们对需求的理解更加完整. 另一方面在多视点间建立了追踪元模型, 并且通过矩阵运算实现了多视点下元模型间的追踪. 这不仅解决了视点分散性问题, 实现了多视点元模型间的统一, 还有效解决了需求变更困难的问题. 然而, 本文在场景元模型和软件体系结构元模型间主要针对功能需求进行了追踪并验证, 缺乏对非功能需求追踪的研究. 接下来的研究方向可从非功能需求在多视点元模型间的追踪继续展开.

参考文献

1 Booch G, Rumbaugh G, Jacobsen I. The unified modeling language user guide. Massachusetts: Addison-Wesley, 1999: 15–17.

2 France R, Bieman JM. Multi-view software evolution: A uml-based framework for evolving object-oriented software. Proceedings of IEEE International Conference on Software Maintenance. Florence, Italy. 2001. 386–395.

3 Leite JCSP, Freeman PA. Requirements validation through viewpoint resolution. IEEE Transactions on Software Engineering, 1991, 17(12): 1253–1269. [doi: 10.1109/32.106986]

4 Kruchten P. Architecture blueprints-the “4+1” view model of software architecture. Proceedings of Tutorial Proceedings on TRI-Ada ’91: Ada’s Role in Global Markets: Solutions for A Changing Complex World. Anaheim, CA, USA. 1995.

5 Finkelstein A, Kramer B, Nuseibeh B, *et al*. Viewpoints: A framework for integrating multiple perspectives in system development. Software Engineering and Knowledge Engineering, 1992, 2(1): 31–57. [doi: 10.1142/S0218194092000038]

6 Cimatti A, Roveri M, Susi A, *et al*. Validation of requirements for hybrid systems: A formal approach. ACM Transactions on Software Engineering and Methodology, 2013, 21(4): Article No.22.

7 Goknil A, Kurtev I, van den Berg K, *et al*. Change impact analysis for requirements: A metamodeling approach. Information and Software Technology, 2014, 56(8): 950–972. [doi: 10.1016/j.infsof.2014.03.002]

8 Tekinerdogan B, Erata F. Modeling Traceability in System of Systems. Proceedings of the Symposium on Applied Computing. Marrakech, Morocco. 2017. 1799–1802.

9 朱雪峰, 金芝. 关于软件需求中的不一致性管理. 软件学报, 2005, 16(7): 1221–1231.

10 王映辉, 王立福, 张世琨, 等. 一种软件需求变化追踪方法. 电子学报, 2006, 34(8): 1428–1432. [doi: 10.3321/j.issn:0372-2112.2006.08.015]

11 Zubcoff JJ, Garrigos I, Casteleyn S, *et al*. Evaluating the use of pareto efficiency to optimize non-functional requirements

satisfaction in i* modeling. IEEE Latin America Transactions, 2016, 14(1): 331–338. [doi: 10.1109/TLA.2016.7430098]

12 周莹新, 艾波. 软件体系结构建模研究. 软件学报, 1998, 9(11): 866–872.

13 Mendonça DF, Rodrigues GN, Ali R, *et al*. GODA: A goal-oriented requirements engineering framework for runtime dependability analysis. Information and Software Technology, 2016, 80: 245–264. [doi: 10.1016/j.infsof.2016.09.005]

14 Roy JF, Kealey J, Amyot D. Towards integrated tool support for the user requirements notation. Proceedings of the 5th International Conference on System Analysis and Modeling: Language Profiles. Kaiserslautern, Germany. 2006.

15 Georg G, Mussbacher G, Amyot D, *et al*. Synergy between activity theory and goal/scenario modeling for requirements elicitation, analysis, and evolution. Information and Software Technology, 2015, 59: 109–135. [doi: 10.1016/j.infsof.2014.11.003]

16 Mylopoulos J, Chung L, Liao S, *et al*. Exploring alternatives during requirements analysis. IEEE Software, 2001, 18(1): 92–96. [doi: 10.1109/52.903174]

17 Gnaho C, Semmak F, Laleau R. Modeling the impact of non-functional requirements on functional requirements. Proceedings of International Conference on Conceptual Modeling. Hong Kong, China. 2017. 59–67.

18 Wei BY, Delugach HS. A framework for requirements knowledge acquisition using UML and conceptual graphs. Lee R. Software Engineering Research, Management and Applications. Cham: Springer, 2016.

19 孙昌爱, 金茂忠, 刘超. 软件体系结构研究综述. 软件学报, 2002, 13(7): 1228–1237.

20 梅宏, 申峻嵘. 软件体系结构研究进展. 软件学报, 2006, 17(6): 1257–1275.

21 Tibermacine C, Sadou S, That MTT, *et al*. Software architecture constraint reuse-by-composition. Future Generation Computer Systems, 2016, 61: 37–53. [doi: 10.1016/j.future.2016.02.006]

22 李潇, 魏长江. 基于追踪矩阵获取完整性需求的研究. 计算机科学, 2019, 46(6): 188–195.

23 王映辉, 张世琨, 刘瑜, 等. 基于可达矩阵的软件体系结构演化波及效应分析. 软件学报, 2004, 15(8): 1107–1115.