UML 顺序图的鲁棒性分析

张　强，　蒋建民，　李建清

(成都信息工程大学 软件工程学院　四川 成都 610225)

摘要：鲁棒性分析技术可以帮助开发人员精确地弥合分析与设计之间的鸿沟。统一建模语言(unified modeling language,UML)顺序图模型被广泛用于面向对象软件系统的分析与设计,它的鲁棒性至关重要。首先,引入形式化模型——统一结构。其次,给出了基于统一结构的描述顺序图的方法和鲁棒性的形式定义,随后讨论顺序图在组合与精化条件下的保存问题。最后,在原型工具支持下进行实例研究。实验结果表明,设计阶段的顺序图能保持分析阶段顺序图的鲁棒性,从而可以给予从事面向对象的开发人员相关帮助及支持。

关键词：UML；顺序图；鲁棒性；精化

中图分类号：TP311.5　　**文献标志码**：A　　**文章编号**：1671-6841(2024)02-0066-07

DOI：10.13705/j.issn.1671-6841.2022244

Robustness Analysis of UML Sequence Diagrams

ZHANG Qiang, JIANG Jianmin, LI Jianqing

(*College of Software Engineering, Chengdu University of Information Technology, Chengdu* 610225, *China*)

Abstract: With the help of robust analysis techniques, developers could precisely bridge the gap between analysis and design. UML (unified modeling language) sequence diagram model was widely used in the analysis and design of object-oriented software systems, and its robustness is of great importance. Firstly, formal model, unified structure, was introduced. Then, a method of describing sequence diagram based on unified structure and a formal definition of robustness were given, and the preservation of sequence diagrams on combination and refinement conditions was discussed. Finally, a case study was carried out with the support of prototype tool. The results showed that the sequence diagram in the design phase could maintain the robustness of the sequence diagram in the analysis phase, which could provide relevant help and support to the object-oriented developers.

Key words: UML; sequence diagram; robustness; refinement

0　引言

在面向对象的软件开发方法中,软件的分析和设计之间存在着断层,无法直接从分析模型平滑地过渡到设计模型[1-3]。统一开发过程(rational unified process,RUP)是最著名的面向对象的开发方法之一[2]。从用例模型导出分析模型,再到设计模型,最后实现编码,需要将用例链接到对象,并进行鲁棒性分析。鲁棒性分析技术可以帮助开发人员弥合从分析到设计之间的鸿沟[2,4]。鲁棒性分析技术是一种分析用例文本,并为每个用例初步确定对象集的方法。这些对象分为边界对象、实体对象和控制器对象。软件系统通过操作或者其他实体与外界通信,增加了“执行者”对象。鲁棒性分析技术就是确定这四类对象关系的技术[1]。

在面向对象的软件开发方法发展过程中,最初的鲁棒性分析技术来自 Jacobson[1],其目的是用于

收稿日期:2022-08-18

基金项目:国家重点研发计划(2022YFB3305104);国家自然科学基金项目(61772004);成都信息工程大学人才科研基金项目(KYTZ202009)。

第一作者:张强(1995—),男,硕士研究生,主要从事软件开发和形式化方法研究,E-mail:1689956075@qq.com。

通信作者:蒋建民(1972—),男,教授,主要从事软件开发和形式化方法研究,E-mail:jjm@fjnu.edu.cn。

分析用例模型的正确性。后来,随着 UML[3] 的出现,Jacobson 采用 UML 协作图作为分析模型,通过分析 UML 协作图的鲁棒性来确定分析模型的鲁棒性,从而达到两者的平滑过渡。然而,顺序图作为面向对象的设计模型,有时也作为分析模型,如何保证从抽象到具体这个过程的鲁棒性,以及如何保证鲁棒的顺序图的正确组合和分析,则很少有人研究。

顺序图是 UML 中重要的交互图之一[4]。它是一种成熟的可视化模型,通过生命线之间的交换消息及消息序列,描述交互关系和发生规范。顺序图模型缺乏精确的形式化定义,无法直接进行形式化验证。

顺序图的鲁棒性主要分析对象之间的关系。本文提出一种称为"统一结构"的形式化模型,从结构方面建模和分析顺序图,可以避免状态爆炸问题。本文用统一结构对顺序图进行建模,给出了鲁棒性的形式定义,研究了分解和组合时如何保证鲁棒性,详细讨论了精化情况下鲁棒性的保持问题,并进行了实例研究和工具介绍。研究结果表明,我们的方法能保证顺序图在软件开发中的鲁棒性。

1　相关工作

鲁棒性分析技术是 RUP[5]、ICONIX[6] 等软件开发方法的重要支撑技术。研究人员也利用鲁棒性分析方法对 UML 模型中存在的问题进行研究。Honda 等认为需求模型和设计模型之间的无缝连接对于完全实现用户需求是非常有效的,同时提出了一种在 ICONIX 中将尽可能多的信息从 KAOS 需求模型转移到系统行为初步设计的方法,以反映模型的需求[7]。Scott 等认为鲁棒性分析方法将分析内容与设计方式联系起来,有助于确保用例文本的正确性[8]。鲁棒图事实上是一种特殊的 UML 通信图,UML 通信图在一定程度上与 UML 顺序图的语义等价,但 UML 顺序图在软件需求分析和设计中有更广泛的用途[9]。

UML 顺序图是 UML 描述动态交互的图,它的形式化语义一直是人们研究的热点和重点。Lund 等根据形式语义不同的表达方式,将顺序图的形式语义分为两大类:操作语义(operational semantics)和指称语义(denotation semantics)[10],它们都是从行为视点出发,分析顺序图的行为,存在状态爆炸的问题。Storrle[11] 最早提出了顺序图基于路径的形式语义,但是对于顺序图中各种组合片段没有精确的定义。Zafar 对 UML 顺序图的组合片段做了相关说明,表示由于 UML 图下隐藏的语义,通过任何计算机辅助的软件工程工具将 UML 模型完全转换为可执行代码的工作并不多,且 UML 模型不能用于系统验证和验证的自动化分析[12]。Chen 等利用 UML 顺序图以增量的形式表达安全关键系统的安全需求[13],对给出模型做一致性检测。姬莉霞等提出了基于时间自动机的 UML 模型之间的转换与验证[14]。另外,我们还没有发现存在任何工作研究顺序图的鲁棒性。

2　统一结构模型

本节先给出用于顺序图建模的形式化模型"统一结构",然后讨论该模型的一些性质。

统一结构(unified structure,US)是我们研究团队在以前依赖结构(dependency structure)[15-16] 的基础上开发的新的基于结构的形式化模型。统一结构的表达能力和兼容性比依赖结构更强,目的是将 UML 模型进行统一的形式化建模,从结构的角度对 UML 模型进行分析。

定义 1　统一结构是一个多元组

$$\langle ME, <, \overset{1}{\omega}, \cdots, \overset{n}{\omega}, \lambda, \overset{1}{\tau}, \cdots, \overset{m}{\tau} \rangle,$$

其中:ME 为模型元素的有限集合;$< \subseteq ME \times ME$ 为包含关系,是一个非自反的偏序;$\overset{i}{\omega} \in ME \times ME$ 为依赖关系,$\forall i \in \{1,\cdots,n\}$;$\lambda \subseteq ME \times (< \cup \overset{1}{\omega} \cdots \cup \overset{n}{\omega})$ 为依赖关系上的限制;$\overset{j}{\tau} \in ME, \forall j \in \{1,\cdots,m\}$ 为模型元素的类型集,它满足条件:$\forall e \in ME, \exists \tau \in \{1,\cdots,m\}: e \in \tau$。

采用在 UML 中相同的模型元素概念[4];包含关系用于建模父子关系,它是非自反的偏序关系;依赖关系可以有多种,不同的模型,种类数量不同;在每个依赖关系中可以存在限制条件;模型元素根据不同视点可以分成不同类型。显然,根据定义 1,可以使用统一结构对顺序图进行建模。

图 1 是一个简单的顺序图,使用统一结构可以表示为

$$SD = \langle ME, <, \lambda, \overset{objClass}{\omega}, \overset{seq}{\omega}, \overset{fragOper}{\omega}, \overset{objInteraction}{\omega}, \overset{ActorObj}{\tau}, \overset{BounObj}{\tau}, \overset{ContObj}{\tau}, \overset{EntiObj}{\tau}, \overset{msg}{\tau}, \overset{class}{\tau}, \overset{operKind}{\tau}, \overset{Fragment}{\tau}, \overset{guard}{\tau} \rangle,$$

其中:$ME = \{actor, Actor, a, b, c, A, B, C, m1, m2, m3, m4, m5, m6, m7, alt, ac1, true, false\}$,表示模型所有元素的集合,$a$ 表示一个对象;

$< = \{(m3, ac1), (m4, ac1), (true, ac1), (false, ac1)\}$,是包含关系,用于描述父子关系,如

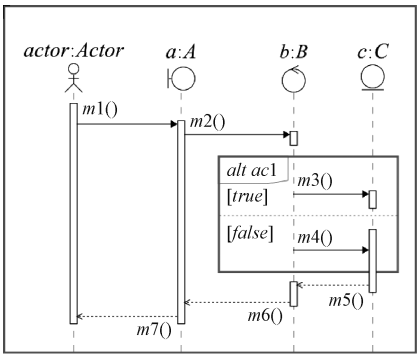

图 1 一个简单顺序图

Figure 1 A simple sequence diagram

$(m3,ac1)$ 表示选择组合框 $ac1$ 包含了元素 $m3$;

$\lambda=\{(m1,(a,actor)),(m2,(b,a)),(m3,(c,b)),(m4,(c,b)),(m5,(b,c)),(m6,(a,b)),(m7,(actor,a))\}$,表示依赖关系上的限制条件,如 $(m1,(a,actor))$ 意味着消息 $m1$ 是由 $actor$ 发送给对象 a;

$\overset{objClass}{\omega}=\{(actor,Actor),(a,A),(b,B),(c,C)\}$,表示对象与类之间的依赖关系,如关系 (a,A) 表示是类的对象;

$\overset{seq}{\omega}=\{(m2,m1),(ac1,m2),(m5,ac1),(m6,m5),(m7,m6),(m3,true),(m4,false)\}$,表示消息出现先后序列,也属于依赖关系,如 $(m2,m1)$ 表示消息 $m1$ 出现后才出现消息 $m2$;

$\overset{fragOper}{\omega}=\{(ac1,alt)\}$,表示组合片段操作的依赖关系,如 $(ac1,alt)$ 表示 $ac1$ 是一个类型组合片段;

$\overset{objInteraction}{\omega}=\{(a,actor),(b,a),(c,b),(b,c),(actor,a)\}$,表示依赖条件的限制关系,如 $(a,actor)$ 表示边界对象 a 依赖执行者对象 $actor$;

$\overset{ActorObj}{\tau}=\{actor\}$,表示执行者对象集合;

$\overset{BounObj}{\tau}=\{a\}$,表示边界对象集合;

$\overset{ContObj}{\tau}=\{b\}$,表示控制器对象集合;

$\overset{EntiObj}{\tau}=\{c\}$,表示实体对象集合;

$\overset{msg}{\tau}=\{m1,m2,m3,m4,m5,m6,m7\}$,表示消息集合;

$\overset{class}{\tau}=\{Actor,A,B,C\}$,表示类型集合;

$\overset{operKind}{\tau}=\{alt\}$,表示图中组合片段元素的类型,如 alt;

$\overset{Fragment}{\tau}=\{ac1\}$,表示组合片段元素的实例,如 $ac1$;

$\overset{guard}{\tau}=\{true,false\}$,表示组合片段的条件元素的集合,如组合框 $ac1$ 的条件 $true$。

定义 2 令 $US=\langle ME,\prec,\overset{1}{\omega},\cdots,\overset{n}{\omega},\lambda,\overset{n}{\tau},\cdots,\overset{m}{\tau}\rangle$ 是一个统一结构,如果 $\forall i\in\{1,\cdots,n-1\}$,$x_i$,$x_{i+1}\in ME$,$(x_i,x_{i+1})\in(\prec\cup\overset{1}{\omega}\cdots\cup\overset{n}{\omega})$ 成立,则可定义序列 $dc=x_1\cdots x_n$ 是结构的依赖链。$\overset{\wedge}{dc}=\{x_1,\cdots,x_n\}$,表示依赖链 dc 中的模型元素。$DC(US)$ 表示结构 US 中所有可能的依赖链。$[dc]$ 表示依赖链 dc 中模型元素的数量,即 $[dc]=n$。如图 1 所示的顺序图模型中,它的一个依赖链子序列为 $dc_1=m2,m1$,表示消息 $m2$ 依赖于消息 $m1$,$\overset{\wedge}{dc_1}=\{m1,m2\}$,表示链中的两个模型元素,此时 $[dc_1]=1$。

命题 1 令 US 为统一结构,且 $dc\in DC(US)$。如果 $\forall i\in\{1,\cdots,n-1\}$,$(x_i,x_{i+1})\in\prec$,那么 dc 中不会存在循环。

证明 由定义 1 可知,$\prec$ 是一种非自反的偏序。并且,由于依赖链 dc 仅有包含关系,dc 中不存在循环。

3 顺序图的鲁棒性

根据文献[1],UML 顺序图的鲁棒性遵守以下规则。

规则 1 执行者只能与边界对象通信。

规则 2 边界对象只能与控制器和执行者通信。

规则 3 实体对象只能与控制器通信。

规则 4 控制器可以与边界对象和实体对象以及其他控制器通信,但不与执行者通信。

根据这四条规则,可以给出以下形式化定义。

定义 3 令 $US=\langle ME,\prec,\overset{1}{\omega},\cdots,\overset{n}{\omega},\lambda,\overset{1}{\tau},\cdots,\overset{m}{\tau}\rangle$,用来构建顺序图。设 $\overset{ActorObj}{\tau}$、$\overset{BounObj}{\tau}$、$\overset{ContObj}{\tau}$、$\overset{EntiObj}{\tau}\in\{\overset{1}{\tau},\cdots,\overset{m}{\tau}\}$,它们分别是该顺序图的执行者、边界对象、控制对象和实体对象的集合,并且设 $\overset{objInteraction}{\omega}\in\{\overset{1}{\omega},\cdots,\overset{n}{\omega}\}$ 是对象交互的关系。如果满足:

$$\forall o_1,o_2\in ME:(o_1,o_2)\in\overset{objInteraction}{\omega}\Rightarrow$$
$$(o_1\in\overset{ActorObj}{\tau}\wedge o_2\in\overset{BoundObj}{\tau})\vee$$
$$(o_2\in\overset{ActorObj}{\tau}\wedge o_1\in\overset{BoundObj}{\tau})\vee$$
$$(o_1\in\overset{BoundObj}{\tau}\wedge o_2\in\overset{ContObj}{\tau})\vee$$
$$(o_2\in\overset{ContObj}{\tau}\wedge o_1\in\overset{BoundObj}{\tau})\vee$$
$$(o_1\in\overset{ContObj}{\tau}\wedge o_2\in\overset{ContObj}{\tau})\vee$$

$$(o_1 \in \overset{BoundObj}{\tau} \wedge o_2 \in \overset{EntiObj}{\tau}) \vee (o_1 \in \overset{ContObj}{\tau} \wedge o_2 \in \overset{EntiObj}{\tau}),$$

那么称 US 是鲁棒的。

定义 3 给出了对象交互限制条件,如图 1 所示,*actor* 是执行者对象,*a* 是边界类对象,*c* 是控制类对象,*b* 是实体类对象,*actor* 通过消息 *m*1、*m*6 与 *a* 交互,*a* 通过消息 *m*2 与 *b* 交互,*b* 通过消息 *m*3、*m*4、*m*5 与 *c* 交互。它们之间的所有通信都符合上述规则,因此在图 1 中的顺序图是鲁棒的。

4　可组合性

简单的顺序图可以通过组合形成复杂的顺序图,接下来研究顺序图在组合操作下的鲁棒性。

定义 4　令 $US' = \langle ME', <', \overset{1}{\omega}', \cdots, \overset{n}{\omega}', \lambda', \overset{1}{\tau}', \cdots, \overset{m}{\tau}' \rangle$ 和 $US'' = \langle ME'', <'', \overset{1}{\omega}'', \cdots, \overset{n}{\omega}'', \lambda'', \overset{1}{\tau}'', \cdots, \overset{m}{\tau}'' \rangle$ 为两个统一结构。如果满足

$$ME' \subseteq ME'', <' \subseteq <'', \forall i \in \{1, \cdots, n\} : \overset{i}{\omega}' \subseteq \overset{i}{\omega}'',$$

$$\forall j \in \{1, \cdots, m\} : \overset{j}{\tau}' \subseteq \overset{j}{\tau}'',$$

称 US' 为 US'' 的子结构,记为 $US' \subseteq US''$。

如果 $<' \cup <''$ 是非自反的偏序,则 US' 和 US'' 之间的组合定义为 $US' \uplus US'' = \langle ME, \overset{1}{\omega}, \cdots, \overset{n}{\omega}, \lambda, \overset{1}{\tau}, \cdots, \overset{m}{\tau} \rangle$,其中:

$$ME = ME' \cup ME''; <= <' \cup <''; \overset{i}{\omega} = \overset{i}{\omega}' \cup \overset{i}{\omega}'',$$

$$\forall i \in \{1, \cdots, n\}; \overset{j}{\tau} = \overset{j}{\tau}' \cup \overset{j}{\tau}'', \forall j \in \{1, \cdots, m\},$$

称 US' 和 US'' 是可组合的。

两个可组合的统一结构可能有不同数量的依赖关系类型及不同的依赖关系。两个统一结构在组合之前将等价转换成相同依赖类型的模型。例如,有两个统一结构分别为

$$US' = \langle ME', <', \overset{a}{\omega}', \lambda', \overset{1}{\tau}' \rangle,$$

$$US'' = \langle ME'', <'', \overset{x}{\omega}'', \lambda'', \overset{1}{\tau}'' \rangle。$$

将 US'、US'' 转换为 US_1 和 US_2,分别为

$$US_1 = \langle ME', <', \overset{a}{\omega}', \overset{x}{\omega}', \lambda', \overset{1}{\tau}' \rangle,$$

$$US_2 = \langle ME'', <'', \overset{a}{\omega}'', \overset{x}{\omega}'', \lambda'', \overset{1}{\tau}'' \rangle,$$

其中:$\overset{x}{\omega}' = \overset{a}{\omega}'' = \varnothing$。

显然,$US_1 = US'$、$US_2 = US'$,且 US_1 和 US_2 具有相同数量的关系类型。因此,根据定义 4,US_1 和 US_2 可以对应相同的依赖类型进行组合。此外,如果模型元素类型集不同,也可采取上述方式进行处理。

性质 1　令 US、US' 和 US'' 是三个统一结构,并且这三个结构之间可以两两组合,则存在

1) $US' \uplus US''$ 是一个统一结构;

2) $US' \uplus US'' = US'' \uplus US'$;

3) $(US \uplus US') \uplus US'' = US \uplus (US' \uplus US'')$。

这些性质都可以直接证明。

这一命题表明统一结构的组合操作具有封闭性、交换性和结合性。

定理 1　令 US 和 US' 是两个统一结构,则

1) 如果 US' 是 US 的一个子结构,且 US 是鲁棒的,则 US' 是鲁棒的;

2) 如果 US 和 US' 是可组合的,且它们是鲁棒的,则 $US \uplus US'$ 是鲁棒的。

证明 1) (反证法)假设 US' 是非鲁棒的,根据定义 1,设 $US' = \langle ME', <', \overset{1}{\omega}', \cdots, \overset{n}{\omega}', \lambda', \overset{1}{\tau}', \cdots, \overset{m}{\tau}' \rangle$,那么必然存在两个对象 o_1'、o_2' 的连接使得该模型是非鲁棒的,如不满足规则 1,执行者只能与边界对象通信,那么有 $o_1' \in \overset{ActorObj}{\tau} \wedge o_2' \in \overset{ContObj}{\tau}'$, $\exists o_1', o_2' \in ME'$,且 US' 是 US 的一个子结构,因此有 $\overset{ActorObj}{\tau} \subseteq \overset{ActorObj}{\tau}'$, $\overset{ContObj}{\tau} \subseteq \overset{ContObj}{\tau}'$,表示 US' 中存在执行对象与控制对象直接交互,这与 US 是鲁棒的相悖,假设不成立,因此 US' 是鲁棒的。同理可证当模型不满足规则 2、规则 3、规则 4 时的情况。

2) 由定义 1,设 $US'' = US \uplus US' = \langle ME'', <'', \overset{1}{\omega}'', \cdots, \overset{n}{\omega}'', \lambda'', \overset{1}{\tau}'', \cdots, \overset{m}{\tau}'' \rangle$,由定义 4 可知 US'' 中包含 US 和 US' 中所有关系和元素且未改变原来的关系,且由于 US 和 US' 都是鲁棒的,因此 US'' 满足定义 3,故 $US \uplus US'$ 是鲁棒的。

5　模型精化

在本文中,假设 ME 是所有模型元素的集合,令 US 表示所有统一结构的集合,令 $DR = ME \times ME$ 为所有包含关系和依赖关系的子集。空的统一结构是指它的模型元素集合、包含关系、依赖关系集合、类型集合都是空集,记为 $\varnothing$。

定义 5　元素精化函数

统一结构的元素精化函数是一个全函数,表示为

$$Eref: ME \rightarrow US \setminus \{\varnothing\}。$$

一个统一结构中的某个模型元素精化为一个统一结构,该统一结构是根据开发者的需要得到的,它

一定不为空。

性质 2 设 $Eref$ 是统一结构的元素精化函数，并设 $a,b \in ME$。设 a、b 的元素精化函数分别为 $Eref(a)=\langle ME_a, \prec_a, \overset{1}{\omega}_a,\cdots,\overset{n}{\omega}_a, \lambda_a, \overset{1}{\tau}_a,\cdots,\overset{n}{\tau}_a\rangle$ 和 $Eref(b)=\langle ME_b, \prec_b, \overset{1}{\omega}_b,\cdots,\overset{1}{\omega}_a, \lambda_b, \overset{1}{\tau}_b,\cdots,\overset{m}{\tau}_b\rangle$。

命题 1 如果 $ME_a \cap ME_b=\varnothing$，则不存在元素 $(x,y) \in ME_a \times ME_b$ 使得 $(x,y) \in (\prec_a \cup \overset{1}{\omega}_a \cdots \cup \overset{n}{\omega}_a) \cup (\prec_b \cup \overset{1}{\omega}_b \cdots \cup \overset{n}{\omega}_b)$ 成立。

证明 因为 $ME_a \cap ME_b=\varnothing$，表示元素 a、b 两者精化后不存在依赖关系，因此该命题成立。

命题 1 表明，如果两个模型元素存在依赖关系，当它们单独精化时，则有可能它们精化后的元素不存在依赖关系，这种情况显然是不正确的，因此必须进行依赖精化。

定义 6 依赖精化函数

令 US 为一个统一结构。令 $Eref$ 为统一结构的元素精化函数。统一结构的依赖精化函数是一个全函数，表示为

$$Dref: ME \to US\setminus\{\varnothing\}。$$

对 $\forall (a,b) \in DR$，有

$$Eref(a)=\langle ME_a, \prec_a, \overset{1}{\omega}_a,\cdots,\overset{n}{\omega}_a, \lambda_a, \overset{1}{\tau}_a,\cdots,\overset{n}{\tau}_a\rangle,$$

$$Eref(b)=\langle ME_b, \prec_b, \overset{1}{\omega}_b,\cdots,\overset{n}{\omega}_b, \lambda_b, \overset{1}{\tau}_b,\cdots,\overset{m}{\tau}_b\rangle,$$

$$Dref(a,b)=\langle ME, \prec, \overset{1}{\omega},\cdots,\overset{n}{\omega}, \lambda, \overset{1}{\tau},\cdots,\overset{m}{\tau}\rangle,$$

其中：

$$ME \in ME_a \cup ME_b;\ \prec \subseteq ME_a \times ME_b;$$

$$\overset{i}{\omega} \subseteq ME_a \times ME_b, \forall i \in \{1,\cdots,n\};$$

$$e \in \overset{j}{\tau} \Rightarrow e \in \overset{j}{\tau}_a \wedge e \in \overset{j}{\tau}_b, \forall j \in \{1,\cdots,m\}, \forall e \in ME。$$

定义 7 令 $US=\langle ME, \prec, \overset{1}{\omega},\cdots,\overset{n}{\omega}, \lambda, \overset{1}{\tau},\cdots,\overset{m}{\tau}\rangle$ 为一个统一结构，其中：$ME=\{e_i,\cdots,e_k\}$；$\prec \cup \overset{1}{\omega}\cdots\cup\overset{n}{\omega}=\{(a_1,a_1'),\cdots,(a_n,a_n')\}$。

令 $Eref$ 为统一结构的元素精化函数，$Dref$ 是统一结构的依赖精化函数。如果 $\forall a,b \in ME$，函数 $Eref(a)$、$Eref(b)$、$Dref(a,b)$ 是可组合的，则 US 的精化定义为

$$ref(US)=Eref(e_1) \uplus \cdots \uplus Eref(e_k) \uplus Dref(a_1,b_1) \uplus \cdots \uplus Dref(a_l,b_l)。$$

通过精化手段可以描述分析模型与设计模型之间的联系。使用统一结构可以将图 2 表示为 SD_{ref}，它是第一节中分析模型精化后的结果，它的各个元素与第一节中的表达相似，其中类 A 精化为 $A1$，类 B 精化为 $B1$、$B2$，类 C 精化为 $C1$、$C2$。SD_{ref} 表示为

$$SD_{ref}=\langle ME, \prec, \lambda, \overset{objClass}{\omega}, \overset{seq}{\omega}, \overset{fragOper}{\omega}, \overset{objInteraction}{\omega}, \overset{ActorObj}{\tau}, \overset{BounObj}{\tau}, \overset{ContObj}{\tau}, \overset{EntiObj}{\tau}, \overset{msg}{\tau}, \overset{class}{\tau}, \overset{operKind}{\tau}, \overset{Fragment}{\tau}, \overset{guard}{\tau}\rangle,$$

其中：

$$ME=\{Actor,actor,a1,b1,b2,c1,c2,A1,B1,B2,C1,C2,m1\_1,m1\_2,m\_21,alt,m\_31,m\_32,ac,m\_41,m\_42,m\_51,m\_52,true,false\};$$

$$\prec=\{(m\_31,ac),(m\_32,ac),(m\_41,ac),(m\_42,ac),(true,ac),(false,ac)\};$$

$$\lambda=\{(m1\_1,(a1,actor)),(m1\_2,(b1,a1)),(m\_21,(b2,b1)),(m\_31,(c1,b2)),(m\_51,(a1,c2)),(m\_52,(actor,a1))\};$$

$$\overset{objClass}{\omega}=\{(actor,Actor),(a1,A1),(b1,B1),(c1,C1),(c2,C2)\};$$

$$\overset{seq}{\omega}=\{(m1\_2,m1\_1),(m\_21,m1\_2),(ac,m\_21),(m\_31,true),(m\_32,m\_31),(m\_41,false),(m\_42,m\_41),m\_51,ac),(m\_52,m\_51)\};$$

$$\overset{objInteraction}{\omega}=\{(a1,actor),(b1,a1),(b2,b1),(c1,b2),(c2,c1),(a1,c2),(actor,a1)\};$$

$$\overset{fragOper}{\omega}=\{(ac,alt)\};\quad \overset{ActorObj}{\tau}=\{actor\};$$

$$\overset{BounObj}{\tau}=\{a1,b1,b2\};\quad \overset{ContObj}{\tau}=\{c1\};\quad \overset{EntiObj}{\tau}=\{c2\};$$

$$\overset{msg}{\tau}=\{m1\_1,m1\_2,m\_21,m\_31,m\_32,m\_41,m\_42,m\_51,m\_52\};$$

$$\overset{class}{\tau}=\{Actor,A1,B1,B2,C1,C2\};$$

$$\overset{operKind}{\tau}=\{alt\};\quad \overset{Fragment}{\tau}=\{ac\};\quad \overset{guard}{\tau}=\{true,false\}。$$

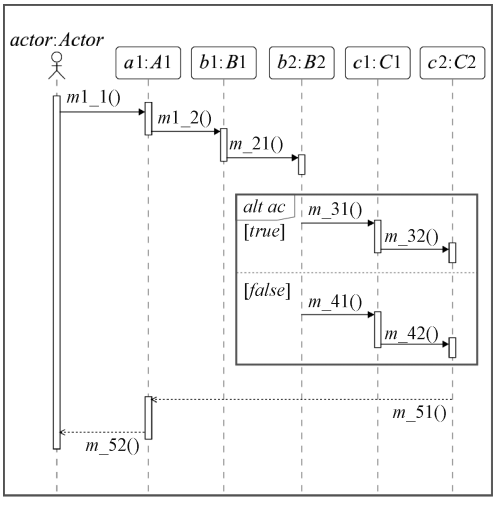

图 2 设计顺序图

Figure 2 The sequence diagram of the design model

定理 2 设 $US=\langle ME, \prec, \overset{1}{\omega},\cdots,\overset{n}{\omega}, \lambda, \overset{1}{\tau},\cdots,\overset{m}{\tau}\rangle$

为一个统一结构,用来描述顺序图,其中 $ME = \{e_1, \cdots, e_k\}$,在顺序图中对象之间的交互关系表示为 $\overset{objInteraction}{\omega} \in (\prec \cup \overset{1}{\omega} \cdots \cup \overset{n}{\omega})$。

令 $Eref$ 为统一结构的元素精化函数,$Dref$ 是统一结构的依赖精化函数。如果 $\forall a, b \in ME$,并且函数 $Eref(a)$、$Eref(b)$ 以及 $Dref(a,b)$ 是可组合的,且都是鲁棒的,则 $Dref(US)$ 是鲁棒的。

证明　由定理 1 可知,两个统一结构可组合且是鲁棒的,则它们的组合是鲁棒的。又根据性质 1,统一结构满足交换律和结合律,那么任意多个满足条件的统一结构,它们的组合是鲁棒的。因此结论成立。

该定理表明,在一个系统的分析与设计中,只要保证每个模型元素和依赖关系在精化过程中是可组合的,并且是鲁棒的,则分析设计得到的系统一定是鲁棒的。

6　实例分析与工具

采用经典案例——自动取款机(ATM)系统进行实验。该系统由取款、转账和存款三个用例组成,这里只采用取款用例进行案例分析。取款用例包含了用户、出纳接口、取款控制、账号和货币分发 5 个类,这些类彼此交互实现用例承担的功能,可以采用 UML 顺序图来描述。分析阶段是设计的初级阶段,它的稳定性在一定程度上决定了软件系统的稳定性。其 UML 顺序图(分析顺序图)使用了“边界类”“控制类”“实体类”的构造类型,统称“分析类”。由于分析类数量较少,分析顺序图的鲁棒性较容易判断,如图 3 所示。

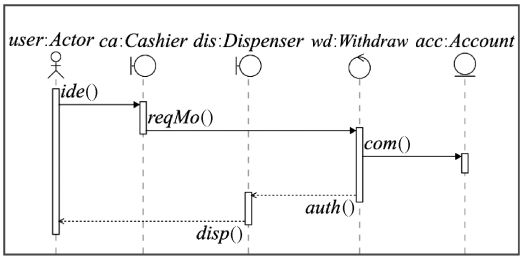

图 3　取款用例的分析顺序图

Figure 3　Analysis sequence diagram of withdrawal case

为获得在设计阶段的顺序图(设计顺序图),需要对其中的各个类型进行精化。通过“跟踪”关系和取款用例的一般细化原则将分析顺序图中的边界类、控制类和实体类进行精化,如表 1 所示。根据表 1 构建取款用例的设计顺序图(图 4)。

表 1　分析模型的精化结果

Table 1　Analysis of model refinement results

分析类	归属类型	精化结果
Cashier	边界类	*CardReader*、*KeyBoard*、*Screen*、*UserManagement*
Dispenser	边界类	*DispenseSenser*、*Contribution*、*CashRegister*
Withdraw	控制类	*Withdraw*、*TaskManagement*
Account	实体类	*Account*、*UserDB*、*AccountManagement*

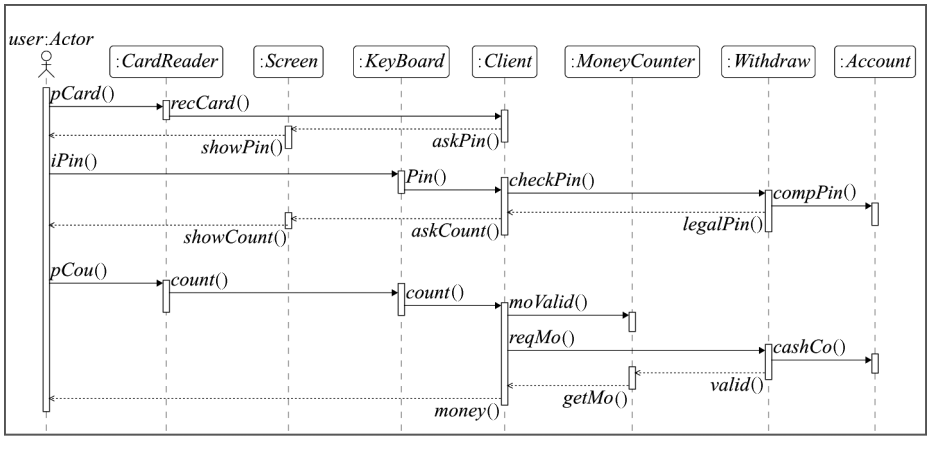

图 4　取款用例的设计顺序图

Figure 4　Design sequence diagram of withdrawal case

为了方便对顺序图模型进行形式化转化和鲁棒性分析,图 3、4 所示的分析、设计顺序图模型均由我们的建模工具绘制。在该建模工具中,一旦一个顺序图构建完成,它能自动转换为统一结构,并进行鲁

棒性分析。利用CASE工具对分析顺序图和设计顺序图进行鲁棒性分析实验,并能判断设计模型是否保持了分析模型的鲁棒性。上述实例的实验结果表明在"跟踪"关系下进行精化操作后,如果分析顺序图模型是鲁棒的,那么其设计顺序图模型也是鲁棒的。

7 结语

UML是一种可视化模型语言,它提供了一套模型概念和图形表示法用于描述软件系统[17]。本文提出了一种UML顺序图的形式化模型,研究了顺序图的鲁棒性及其保持问题。在此过程中,对该形式化模型的可组合性和模型细化等相关性质进行定义和说明,利用这些性质,可以判断在软件开发过程中分析阶段和设计阶段所产生的UML顺序图模型的鲁棒性是否得到了保持。实验结果表明,本文结论是正确的。但本文关注的是UML顺序图的鲁棒性,对于分析模型和设计模型两者的一致性未严格要求。在未来的工作中,将深入研究UML顺序图的一致性与鲁棒性的关系,探索基于UML顺序的代码自动生成方法。

参考文献:

[1] JACOBSON I. Use cases—yesterday, today, and tomorrow[J]. Software & systems modeling, 2004, 3(3): 210-220.

[2] JACOBSON I, BYLUND S, FIRESMITH D G. The Road to the unified software development process[M]. New York: Cambridge University Press,2000.

[3] JACOBSON I, SPENCE I, SEIDEWITZ E. Industrial-scale agile: from craft to engineering[J]. Communications of the ACM, 2016, 59(12): 63-71.

[4] BOOCH G, RUMBAUGH J, JACOBSON I. Unified modeling language user guide[M]. New Jersey: Addison-Wesley Professional, 2005.

[5] KRUCHTEN P. The rational unified process: an introduction[M]. New Jersey: Addison-Wesley Professional, 2003.

[6] ROSENBERG D, STEPHENS M, COLLINS-COPE M. Agile development with ICONIX process[M]. New York: Apress, 2005.

[7] HONDA K, NAKAGAWA H, TAHARA Y, et al. Goal-oriented robustness analysis[J]. Frontiers in artificial intelligence and applications, 2012, 240: 171-180.

[8] SCOTT K, ROSENBERG D. Successful robustness analysis[EB/OL]. [2022-06-21]. http://www.dca.fee.unicamp.br/cursors/EA976/Referencias/sd/sra.html.

[9] 郭艳燕, 张楠, 童向荣. UML顺序图形式化语义的研究综述[J]. 计算机科学, 2017, 44(2): 17-30, 64.
GUO Y Y, ZHANG N, TONG X R. Survey on formal semantics of UML sequence diagram[J]. Computer science, 2017, 44(2): 17-30, 64.

[10] LUND M S, REFSDAL A, STØLEN K. 4 semantics of UML models for dynamic behavior[M]. Berlin: Springer Press, 2010.

[11] STORRLE H. Semantics of interactions in UML 2.0[C]// IEEE Symposium on Human Centric Computing Languages and Environments. Piscataway: IEEE Press, 2003: 129-136.

[12] ZAFAR N A. Formal specification and verification of few combined fragments of UML sequence diagram[J]. Arabian journal for science and engineering, 2016, 41(8): 2975-2986.

[13] CHEN X H, LIU Q, MALLET F. Formally verifying consistency of sequence diagrams for safety critical systems[J]. Science of computer programming, 2022, 216: 102777.

[14] 姬莉霞, 马建红. 基于时间自动机的UML模型转换与验证研究[J]. 郑州大学学报(理学版), 2013, 45(1): 50-55.
JI L X, MA J H. Research on model transformation and model checking of UML based on timed automata[J]. Journal of Zhengzhou university (natural science edition), 2013, 45(1): 50-55.

[15] JIANG J M, ZHU H B, LI Q, et al. Isolation modeling and analysis based on mobility[J]. ACM transactions on software engineering and methodology, 2019, 28(2): 1-31.

[16] JIANG J M, ZHU H B, LI Q, et al. Event-based functional decomposition[J]. Information and computation, 2020, 271: 104484.

[17] 王黎明,柴玉梅. UML中的关联关系及其实现模式[J]. 郑州大学学报(理学版),2002,34(3):25-28.
WANG L M, CHAI Y M. Association in UML and its implementation pattern[J]. Journal of Zhengzhou university (natural science edition), 2002, 34(3):25-28.