

一种 UML 模型跨图一致性检查系统设计与实现

袁 薇¹ 孙康健¹ 张 雪¹ 吕 韬¹
YUAN Wei SUN Kangjian ZHANG Xue LÜ Tao

摘 要

统一建模语言是获取系统需求、进行系统设计的重要手段,使用多种 UML 模型图对同一软件进行描述和表达时,多个模型图中的对象命名、逻辑关系可能出现不一致的情况。现有可视化 UML 建模工具大多只在某一种模型图内部进行正确性检查,或依赖于人工进行跨图的一致性检查。随着软件系统规模的不断增大和逻辑复杂性的增高,仅依赖于人工难免存在疏漏。对此,设计并实现了一种基于规则的 UML 模型跨图一致性检查系统。用户上传的模型文件被解析后,采用所提出的 7 条规则,通过 OCL 描述对上述组件进行模型分析和跨图一致性检查。其中,模型分析包括词法分析、逻辑结构分析等,一致性检查规则包括类图、状态图、顺序图的相关规则。通过实现与验证,所设计的系统能有效地检测类图、顺序图等图中的不一致问题。

关键词

UML; 可视化建模; 模型解析; 一致性检查; 对象约束语言

doi: 10.3969/j.issn.1672-9528.2024.10.008

0 引言

统一建模语言(Unified Modeling Language, UML)是一种用于软件开发的标准化图形建模语言,是获取系统需求、进行系统设计的重要手段,也是后续进行软件开发和实现等工作的依据^[1-2]。对于同一软件系统,用户可能使用多种 UML 模型图进行描述和表达,如类图、用例图、顺序图等,这些模型图大多由人工设计,并且可能由不同的人员完成,因此,多个模型图中的对象命名、逻辑关系可能出现不一致的情况。此外,UML 模型中可能由于缺乏关于对象的附加约束而产生歧义^[3-8]。

现有的可视化 UML 建模工具中,大多只在某一种模型图内部进行正确性检查,若具有针对同一系统的多种模型图,通常只能依赖于人工进行跨图的一致性检查^[9-12]。随着软件系统规模的不断增大和逻辑复杂性的增高,仅依赖于人工难免存在疏漏,因此,本文设计并实现了一种基于规则的 UML 模型一致性检查系统。

1 基于规则的 UML 模型跨图一致性检查系统架构

为了进行模型一致性检测,本文设计的系统将从用户上传的 UML 模型文件中提取类图、状态图、顺序图中的相关要素,并通过模型元素名称及模型逻辑结构分析推断出模型

文件中的跨图不一致性问题,系统架构如图 1 所示。

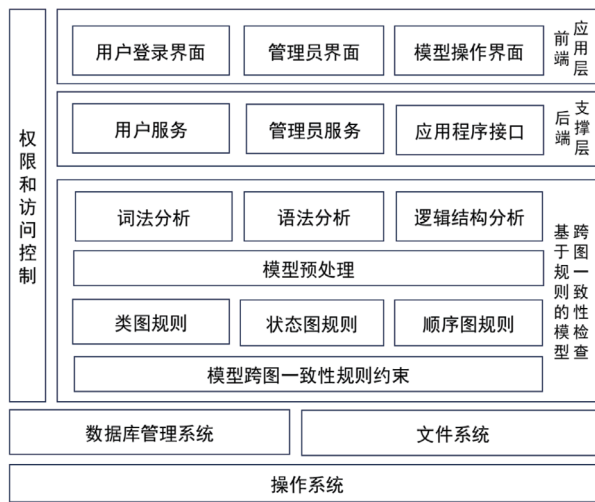


图 1 基于规则的 UML 模型跨图一致性检查系统架构

(1) 系统底层架构

一致性检查系统采用 LAMP (即 Linux-Apache-MySQL-PHP) 架构进行部署。系统底层包括 Linux 操作系统、数据库管理系统(使用关系型数据库 MySQL)、文件系统,以及 Tomcat、Apache 网络服务器。数据库管理系统上配置了 LAMP 运行环境及 Magento 库,具有完整的用户请求响应功能。

(2) 系统权限和访问控制

系统权限和访问控制主要通过 Magento 服务器实现,Magento 在关系型数据库中建立了权限表,记录所有类型用

1. 中国软件评测中心(工业和信息化部软件与集成电路促进中心) 北京 100048

户的权限和允许访问的内容。当用户访问受限数据时,需提供相应的口令。系统中每个模块也具有相应的访问控制配置文件,确保每个应用接口的调用者都具有相应的权限。

(3) 基于规则的模型一致性检查

基于规则的模型一致性检查模块是本系统的核心,其中,位于底层的是模型一致性规则约束子模块,位于上层的是模型分析子模块。模型一致性规则包括类图规则、状态图规则、顺序图规则等,模型分析包括词法分析、逻辑结构分析等,上述模块通过 SOAP 和 RESTful 两种类型的接口提供服务。

(4) 系统后端支撑层

系统后端支撑层主要提供用户服务、管理员服务以及应用程序接口。应用程序编程接口接受用户请求后对其进行处理,然后将具体参数转发至模型一致性检查模块的接口做进一步检查和分析。

(5) 系统前端应用层

系统前端应用层主要包括用户登录界面、管理员界面以及模型操作界面,用户登录界面支持用户登录、登出、密码重置等基本操作,管理员界面支持对用户和系统的管理。

2 UML 模型文件解析

2.1 模型文件结构

本文提出的模型一致性检查系统基于 StarUML 中的模型

文件实现。StarUML 将模型文件(.mdj 文件)表示为树状结构的 JSON 字符串。在 JSON 树状结构中,每一层数据都存储在一个数组中,数组的每一个元素都是一个“键”-“值”对。其中,“值”也是 JSON 字符串,因此可以递归地进行解析。可展开的模型元素树包含了模型文件中的所有信息。

如图 2 所示,一个模型文件可能包含若干个元素,可能是 UML 模型(UML model)、UML 合作(UML collaboration)、UML 状态机(UML state machine)或其他 UML 元素,如活动图、用例图等。为检查模型一致性,需要逐层对这些元素进行解析。一个 UML 模型中可能嵌套其他所有类型的元素,包括 UML 模型。UML 合作中可能嵌套其他合作,但至少包含一个 UML 交互(UML interaction)。UML 合作中也可能包含 UML 状态机,但不能包含 UML 模型。UML 模型可能包含类图,类图元素包括类(class)、枚举(enumeration)、原始类型(primitive type)、信号(signal)、数据类型(data type),UML 类中保留着泛化(generalization)、关联(association)、依赖(dependency)关系等信息。UML 状态机中通常包含若干个状态图(UML Statechart Diagram)以及相应的 UML 区域(UML region)。每个 UML 区域中可能含有转移、状态以及伪状态(如分支、汇合、选择等)。UML 状态中可能会嵌套子区域,用于表示组合状态,且组合状态中仍然可能含有状态、转移或子区域。

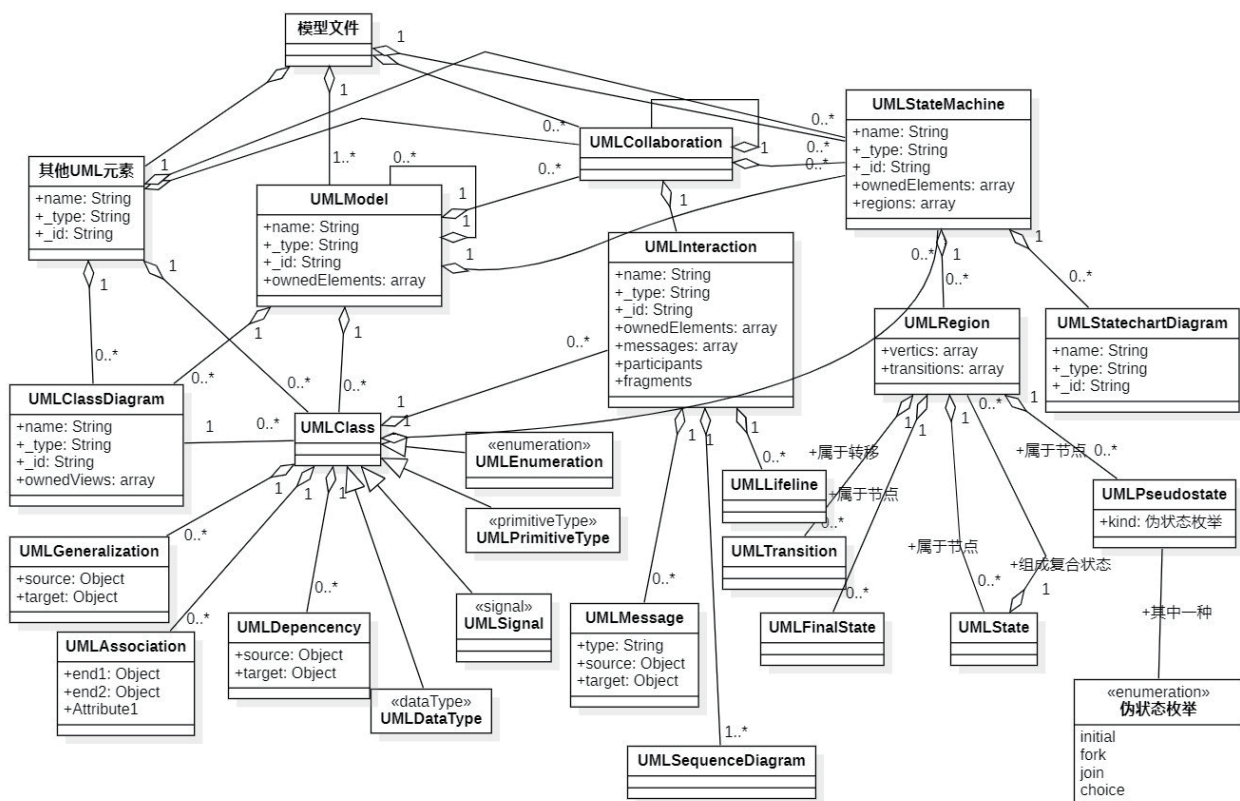


图 2 UML 模型文件结构

2.2 模型文件解析

模型文件可能含有复杂的嵌套关系，因此，按照图 3 所示的状态机对用户上传的模型文件进行解析。首先，将模型文件转换为 JSON 数组格式。若树状结构的根节点下包含 UML 模型，则优先解析 UML 模型；若根节点下包含未定义的模型解析类型，如 UML 用例图、UML 活动图、UML 组件图等，则递归解析其内部结构，并提取与类图、顺序图、状态图相关的组件；若 UML 模型中包含 UML 类或 UML 合作，则分别进行解析；若 UML 合作中包含 UML 交互或 UML 状态机，则分别进行解析。UML 状态机中只可能包含 UML 状态图，所以无需再递归解析其他类型。需要注意的是，UML 类中可能包含 UML 模型中可能出现的所有元素，UML 合作中也可能嵌套 UML 合作，因此都需要进行递归解析。最后，将所有分支结果进行整合，作为后续模型正确性及一致性的解析数据。

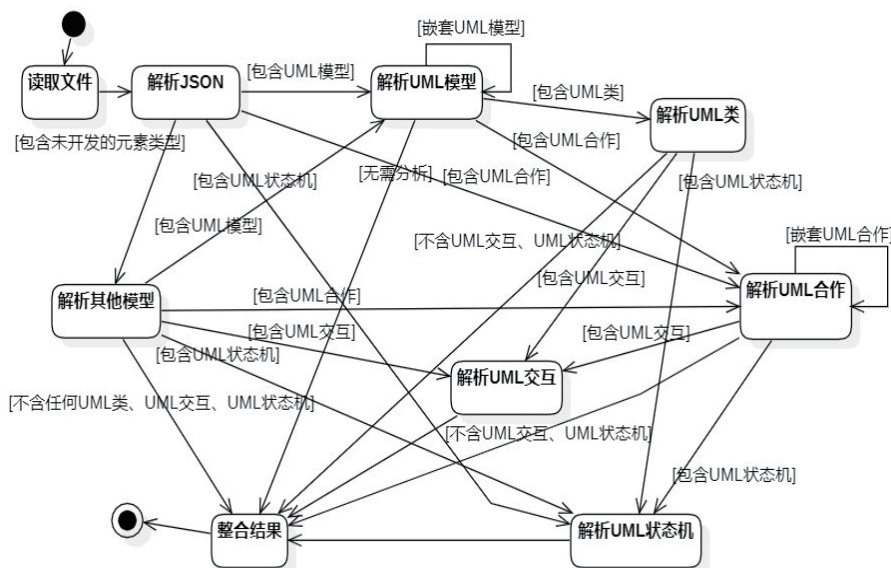


图 3 模型文件解析状态机

3 UML 模型跨图一致性检查规则

模型文件被解析后，可以获得本文所关注的类图组件、顺序图组件以及状态图组件。对象约束语言（object constraint language，OCL）是一种用于表达 UML 模型约束的形式化语言，它允许开发人员在 UML 模型中定义更加精细的规则和约束，以确保模型的一致性和完整性。为满足跨图一致性，本文定义了 7 条 UML 模型跨图一致性检查规则，并通过 OCL 描述对上述组件进行检查。

规则 1：多个类图中相同对象的继承关系必须相同。

输入数据：类图集合。

OCL 描述：

context ClassDiagram

inv: self.R1->forAll(

c1,c2,c3,c4: Class,

d1,d2: ClassDiagram

and c1,c2 includes d1

and c3,c4 includes d2 |

c1.name=c2.name and

c3.name=c4.name implies if

c1 includes c2.generalization

then

c3 includes

c4.generalization

else

if

c1 excludes c2.generalization

then

c3 excludes

c4.generalization

endif

endif

规则 2：多个类图中相同对象的组

合关系必须相同。

输入数据：类图集合。

OCL 描述：

context ClassDiagram

inv: self.R1->forAll(

c1,c2,c3,c4: Class,

d1,d2: ClassDiagram

and c1,c2 includes d1

and c3,c4 includes d2

|

c1.name=c2.name and

c3.name=c4.name implies if

c1 includes c2.composition

then

c3 includes

c4.composition

else

if

c1 excludes c2.composition

then

c3 excludes

c4.composition

endif

endif

规则 3： 多个类图中相同对象的聚合关系必须相同。

输入数据：类图集合。

OCL 描述：

```
context ClassDiagram
inv: self.R1->forAll(
c1,c2,c3,c4: Class,
d1,d2: ClassDiagram
and c1,c2 includes d1
and c3,c4 includes d2 |
c1.name=c2.name and
c3.name=c4.name implies if
    c1 includes c2.aggregation
then
    c3 includes
    c4.aggregation
else
    if
        c1 excludes c2.aggregation
    then
        c3 excludes
        c4.aggregation
    endif
endif
```

规则 4： 多个类图中相同对象的依赖关系必须相同。

输入数据：类图集合。

OCL 描述：

```
context ClassDiagram
inv: self.R1->forAll(
c1,c2,c3,c4: Class,
d1,d2: ClassDiagram
and c1,c2 includes d1
and c3,c4 includes d2 |
c1.name=c2.name and
c3.name=c4.name implies if
    c1 includes c2.dependency
then
    c3 includes
    c4.dependency
else
    if
        c1 excludes c2.dependency
    then
        c3 excludes
```

c4.dependency

endif

endif

规则 5： 顺序图的消息必须在对应的类图中可见，即如果某个顺序图中的对象发送的消息为对应类的方法，那么该方法必须被预先定义。

输入数据：顺序图消息、类图方法。

OCL 描述：

```
context ClassDiagram, SequenceDiagram
inv: self.R1->forAll(
msg:message |
msg.name includes self.R1. operations)
```

规则 6： 顺序图的消息不能违背访问控制权限，即只有继承者可以访问保护方法和属性，私有操作只能被对象自己访问，顺序图中的消息不能违背访问权限。

输入数据：顺序图、类图。

OCL 描述：

(1) 消息对象必须与对应类名一致。

```
context ClassDiagram, SequenceDiagram
```

```
inv: self.R1->forAll(
msg:message, l1: participants, l2: participants, c1:Class, c2:-
```

Class

```
and msg.source = l1
```

```
and msg.target = l2
```

```
and l1.name = c1.name
```

```
and l2.name = c2.name |
```

```
msg.name includes c2. operations)
```

(2) 消息对应的方法不能是私有的。

```
context ClassDiagram, SequenceDiagram
```

```
inv: self.R1->forAll(
```

```
msg:message, l1: participants, l2: participants, c1:Class, c2:-
```

Class

```
and msg.source = l1
```

```
and msg.target = l2
```

```
and l1.name = c1.name
```

```
and l2.name = c2.name |
```

```
msg.name includes c2.none_private_operations)
```

(3) 消息对应的非继承关系的方法不能是受保护的。

```
context ClassDiagram, SequenceDiagram
```

```
inv: self.R1->forAll(
```

```
msg:message, l1: participants, l2: participants, c1:Class, c2:-
```

Class

```
and msg.source = l1
```

```
and msg.target = l2
```



```

and l1.name = c1.name
and l2.name = c2.name |
msg.name includes
c2.none_protected_operations implies c1 includes
c2.generalization)

```

规则 7: 顺序图中的对象必须在类图中可见, 即顺序图中的时间线名称必须与对应类图的类名一致。

输入数据: 顺序图、类图。

OCL 描述:

```
context ClassDiagram, SequenceDiagram
```

```
inv: self.R1->forAll(
```

```
l: participants |
```

```
l.name includes self.R1.classes)
```

4 UML 模型跨图一致性检查系统实现

进行系统实现时, UML 模型跨图一致性检查系统对应的类图如图 4 所示, 主要包括用户账户管理模块、文件管理模块、一致性检查模块、规则模块等。其中, 一致性检查模块中包括对词法、语法以及逻辑结构的检查。

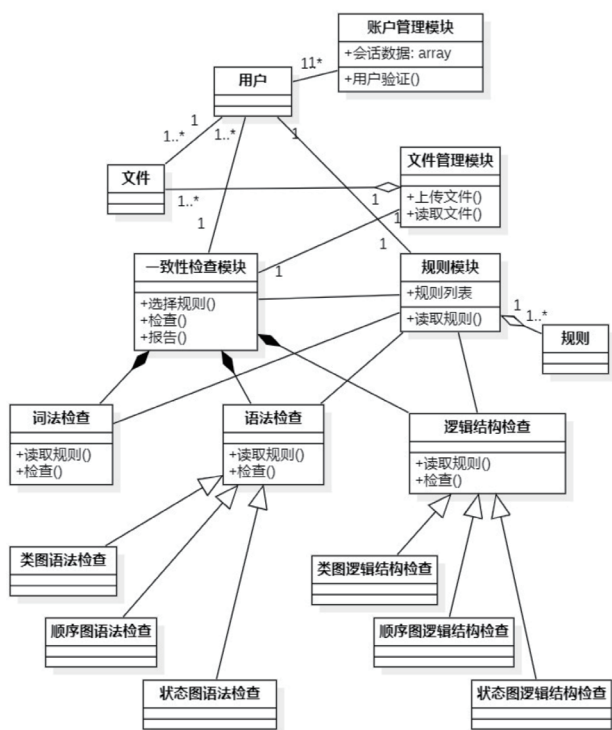


图 4 系统类图

5 结语

针对不同 UML 模型图之间可能存在的 inconsistency 问题, 本文设计并实现了一种基于 OCL 约束的 UML 模型跨图一致性检查系统。首先, 对用户上传的模型文件进行解析; 然后, 针对解析后的组件和元素的名称及逻辑关系, 使用 OCL 规则

进行跨图一致性解析, 从而有效地检测类图、顺序图等图中的不一致问题。

参考文献:

- [1] 田丽从, 周伯生. UML 可视化建模工具中模型一致性检查机制的研究与实现 [J]. 计算机应用与软件, 2005, 22(1): 24-26.
- [2] 王云, 刘又诚, 周伯生. UML 可视化建模系统的模型一致性检查机制 [J]. 计算机研究与发展, 2000, 37(1): 1-8.
- [3] 曾一, 李函逾, 刘慧君, 等. UML 模型和 Java 代码之间的一致性检测方法 [J]. 计算机科学, 2015, 42(4): 151-155.
- [4] 李明宇. UML 模型一致性检测的研究 [D]. 济南: 山东大学, 2015.
- [5] 朱晨. UML 模型一致性检测的研究与设计 [D]. 上海: 上海交通大学, 2007.
- [6] 陈丹, 窦万峰. UML 顺序图与状态图的一致性检查 [J]. 计算机工程, 2008, 34(18): 62-64.
- [7] 董庆超, 王智学, 张爱辉, 等. 基于 UML 类图模型的一致性检查方法 [J]. 计算机技术与发展, 2008, 18(10): 85-88.
- [8] 许培嘉, 赵合计. 基于多个域的 UML 模型的一致性检测 [J]. 计算机工程与设计, 2005, 26(5): 1355-1372.
- [9] 李函逾. 基于模型的代码一致性检测方法的研究 [D]. 重庆: 重庆大学, 2015.
- [10] 王雷. 基于图论和 FSM 的 UML 模型与代码一致性检测 [J]. 北京邮电大学学报, 2019, 42(4): 70-75.
- [11] EGYED A. Instant consistency checking for the UML [C]// Proceedings of the 28th International Conference on Software Engineering. New York, NY: ACM, 2006: 381-390.
- [12] KRISHNAN P. Consistency checks for UML [C]// Proceedings of the 17th Asia-Pacific Software Engineering Conference. New York, NY: ACM, 2000: 162-169.

【作者简介】

袁薇 (1990—), 女, 河南许昌人, 博士, 工程师, 研究方向: 软件缺陷测试与分析。

孙康健 (1982—), 男, 内蒙古通辽人, 本科, 工程师, 研究方向: 信息技术产品测评。

张雪 (1993—), 女, 山东聊城人, 硕士, 工程师, 研究方向: 软件测试。

吕韬 (1987—), 男, 山西临汾人, 硕士, 高级工程师, 研究方向: 基础软件测试技术。

(收稿日期: 2024-07-02)