文章编号:1674 - 2869(2011)09 - 0101 - 04

一种基于 KAOS 和 XML 的横切关注点识别方法

涂成茂,何成万
(武汉工程大学 计算机科学与工程学院,湖北 武汉 430074)

摘 要:提出了基于自动规范中的知识获取 KAOS 和可扩展标记语言 XML 的识别方法,该方法首先对待开发系统进行需求建模,然后将需求模型映射于可扩展标记语言 XML 文件中,最后给出横切关注点识别算法,该算法在案例系统中的实验结果表明此方法的可行性和稳定性.

关键词:横切关注点识别;面向方面需求工程;自动规范中的知识获取;可扩展标记语言
中图分类号:TP311.5 **文献标识码**:A **doi**:10.3969/j.issn.1674-2869.2011.09.025

0 引 言

横切关注点会造成交织和分散的问题,降低了软件的可读性、可维护性和可复用性.关注点分离作为软件开发中重要准则之一,软件系统一般由多个关注点组成,其中包括核心级关注点和系统级关注点.核心级关注点指与业务逻辑相关的主过程,系统级关注点是指业务逻辑之外的过程,如日志,安全,性能等.而某些系统级关注点会表现出横切行为,即一些核心级关注点需要使用若干个系统级关注点提供的服务,同时某些系统级关注点也影响多个核心级关注点的状态[1].具有横切行为的关注点称为横切关注点,横切关注点有两个重要特点:

(1)交织(Tangling):横切关注点参与某关注点的实现

(2)分散(Scattering):横切关注点分散在多个关注点之中

横切关注点妨碍了计算机软件设计中的低耦合[2],面向方面需求工程(Aspect Oriented Requirement Engineering,AORE)[3]的主要目标是需求阶段横切关注点的分离.在目前 AORE 提出的各种方法中,有的只能识别非功能性需求而具有局限性[4],基于 Petri 网的方法尚不能表明运用于大中型系统中[5].

本文提出一种基于 KAOS(knowledge acquisition in automated specification)和 XML(extensible markup language)的横切关注点识别方法.首先利用 KAOS 中的目标模型(Goal Model)在不同层次上对待开发系统进行需求建模.高层次上将待开发系统分解为各个子系统,以保证对横切关注点进行系统的识别;低层次上将高层次目标精细化,然后将需求模型存储于 XML 文件,最后利用 XQuery 实现的横切关注点识别算法完成对横切关注点的识别.

1 横切关注点的识别方法及应用

横切关注点的识别分 3 步:建立目标模型、存储目标模型、横切关注点识别.本文将以某酒店管理系统中住宿管理子系统[5]来详细描述横切关注点的识别过程.该子管理系统的需求如下所述:

(1)顾客请求预定房间:在接受预定前,首先检查是否有空的房间,若有空房间则分配房间并创建预定记录,接受顾客预定请求,否则拒绝顾客请求.

(2)顾客请求入住酒店:顾客在请求入住时,检查顾客是否有预定房间,若有则消除预定记录;若没有则检查是否有空房间,若有则为顾客分配房间,接受顾客入住请求并为顾客创建账单,否则拒绝顾客入住请求.

(3)顾客请求离开酒店:顾客在请求离开前,首先检查顾客是否有入住记录和未付费的账单,若有则计算账单、为顾客结账、消除入住记录,同时接受顾客离开请求.

(4)系统需提供日志功能:系统中任何操作需要被记录并保存.

1.1 建立需求目标模型

根据待开发系统的需求描述运用 KAOS 建立

收稿日期:2011 - 05 - 25

作者简介:涂成茂(1987 -),男,湖北洪湖人,硕士研究生.研究方向:软件工程.
指导老师:何成万,男,教授,博士,硕士研究生指导老师.研究方向:软件工程.

目标模型(Goal Model).

目标是 KAOS 中的重要概念,KAOS 是需求工程中建立需求模型的方法学[6]. 在 KAOS 建立的层次化目标模型中,除了最顶层的战略目标外每个层次中的目标都能被上层目标证明其存在的合理性,也就是为什么该目标存在于模型中;同时除最底层的叶子目标外每个层次中的目标都能被一组子目标精细化,也就是应该怎样实现此目标. 通过将待开发系统的战略目标精细化为各层次子目标,当子目标精细化到其责任能够被分配到主体(Agent)时,就得到了目标模型,此时最底层的叶子目标也称为不可精细化的子目标.

根据住宿管理子系统需求描述而建立的基于 KAOS 的目标模型(Goal Model)如图 1 所示.

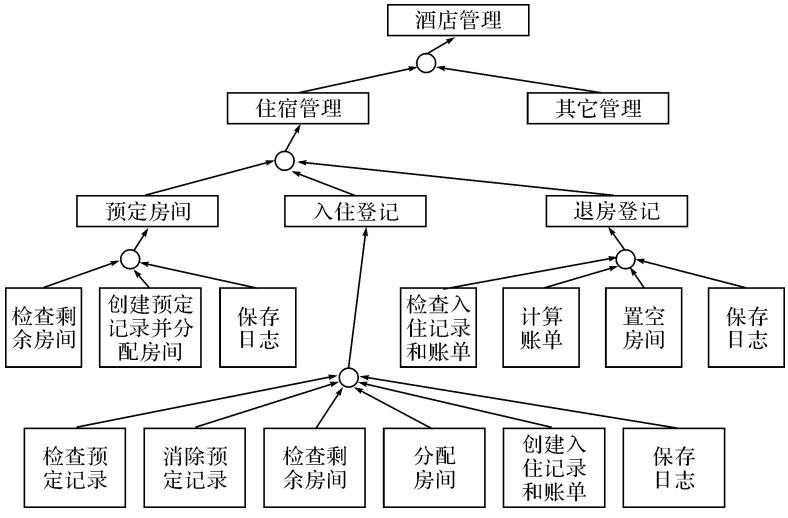

图 1 酒店管理系统的目标模型

Fig. 1 Goal Model of Hotel Manager System

1.2 存储目标模型

在此步中需将目标模型映射至 XML 文件中. XML 是一种元标记语言,是一套可以自定义特定领域语义标记的规范,这些标记将文档分成许多部件并对这些部件加以标识,利用 XML 可以方便描述对象及其关系[7].

首先将 Goal Model 层次结构映射至 XML 模式文件中,然后根据 XML 模式文件将 Goal Model 模型完整映射至 XML 文件中. 在将 Goal Model 层次结构映射至 XML 模式文件时,须依据关注点流、关注点、元关注点的语义创建 XML 元标记,其中包括标记名称和类型,其映射关系如表 1 所示,根据表中的自定义元标记而生成的 HotelManager. xml 文件如图 2 所示,其中存储了酒店管理系统需求模型.

1.3 横切关注点识别

在此步中,将依据横切关注点的定义给出横切关注点识别算法.

1.3.1 *横切关注点定义* 依据目标模型中各层次目标语义,现给出关注点及横切关注点等概念定义:

(1)关注点:目标模型中任一层次中一个特定的目标.

(2)元关注点:目标模型中自上而下精细化过程中不可精细化的子目标.

(3)关注点流:目标模型中某目标被精细化后的子目标集合,规定元关注点的关注点流为其自身.

(4)横切关注点:在目标模型同层次中,任意两两关注点的关注点流交集中的子目标.

(5)关注点扩散度:关注点在目标模型中出现的次数.

表 1 目标模型与 XML 元标记映射关系

Table 1 Mapping between Goal Model and XML Metaconcern

元标记	Goal Model
＜StrategicalGoal/＞	酒店管理
＜KernaGoal/＞	住宿管理、其它管理
＜Concern/＞	预定房间、入住登记、退房登记
＜ConcernFlow/＞	{检查剩余房间、创建预定并分配房间、保存日志}、{…}、…
＜MetaConcern/＞	检查剩余房间、检查预定记录、计算账单、保存日志、…

1.3.2 *横切关注点识别算法* 根据横切关注点定义,则显然有如下结论:关注点是横切关注点与关注点的扩散度至少大于 1 互为充要条件,推导如下:

(1)充要性:若关注点 Concern 为横切关注点,则根据横切关注点定义,C1 必然出现于至少 2

个关注点的关注点流中，由此可以推出 $C1$ 的关注点扩散度必然大于 1.

(2)必要性：若关注点 Concern 的扩散度至少大于 1，则根据扩散度的定义，该关注点必然出现于至少 2 个不同的关注点流 aConcernFlow 与 bConcernFlow 中，令与 aConcernFlow 与 bConcernFlow 相对应的关注点分别为 aConcern 和 bConcern，因此 aConcern 与 bConcern 的关注点流交集中至少包含 Concern，因此 Concern 为横切关注点.

根据以上结论，有如下横切关注点识别算法：

(1)取 3 个目标集合 $C1$，$C2$，$C3$ 并置为空，$C3$ 用于置入横切关注点；

(2)将需求模型中的目标逐一划入集合 $C1$；同时将其逐一划入集合 $C2$，但对于重复元素只放入一次；

(3)计算 $C2$ 中各个元素的扩散度. 遍历 $C2$ 集合中的元素，并计算该元素在 $C1$ 集合中出现的次数；

(4)将扩散度大于 1 的目标划入横切关注点集合 $C3$；

算法所识别的横切关注点存在于 $C3$ 中.

```
<StrategicalGoal xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="HotelManager.xsd">
    <StrategicalGoalName>酒店管理</StrategicalGoalName>
    <KernaGoals>
        <KernaGoal>
            <KernaGoalName>住宿管理</KernaGoalName>
            <Concern>
                <ConcernName>预定房间</ConcernName>
                <ConcernFlow>
                    <MetaConcern>检查剩余房间</MetaConcern>
                    <MetaConcern>创建预定记录并分配房间</MetaConcern>
                    <MetaConcern>保存日志</MetaConcern>
                </ConcernFlow>
            </Concern>
            <Concern>
                <ConcernName>入住登记</ConcernName>
                <ConcernFlow>
                    <MetaConcern>检查预定记录</MetaConcern>
                    <MetaConcern>消除预定记录</MetaConcern>
                    <MetaConcern>检查剩余房间</MetaConcern>
                    <MetaConcern>分配房间</MetaConcern>
                    <MetaConcern>创建入住记录和账单</MetaConcern>
                    <MetaConcern>保存日志</MetaConcern>
                </ConcernFlow>
            </Concern>
            <Concern>
                <ConcernName>退房登记</ConcernName>
                <ConcernFlow>
                    <MetaConcern>检查入住记录和账单</MetaConcern>
                    <MetaConcern>计算账单</MetaConcern>
                    <MetaConcern>消除入住记录和账单</MetaConcern>
                    <MetaConcern>置空房间</MetaConcern>
                    <MetaConcern>保存日志</MetaConcern>
                </ConcernFlow>
            </Concern></KernaGoal>
    </KernaGoals></StrategicalGoal>
```

图 2　XML 中的目标模型

Fig. 2　Goal Model in XML

1.4　方法应用及比较

此节给出算法实现、案例应用以表明方法的可行性和稳定性.

1.4.1　*算法实现*　　运用 1.3 节中算法思想并基于 XQuery 的算法实现如图 3 所示，其中行 5 定义了元关注点层次上去除重复元素后的关注点集合 $C2$，行 7 统计了 $C2$ 中各元素在文档中出现的次数，行 8 设定了选入 $C3$ 集合的条件，行 10 至行 13 定义了 $C3$ 集合中的元素及其扩散度.

```
1   xquery version "1.0";
2   <result>
3   {
4       let $doc:=doc("HotelManager.xml")
5       for $metaconcern at $i in distinct-values($doc//MetaConcern)
6       let $metaconcerntext:=data($metaconcern)
7       let $dispersity:=count($doc//MetaConcern[text()=$metaconcerntext])
8       where $dispersity>1
9       return
10          <CrosscuttingConcern>
11              <MetaConcern>{$metaconcern}</MetaConcern>
12              <Dispersity>{$dispersity}</Dispersity>
13          </CrosscuttingConcern>
14  }
15  </result>
```

图 3　基于 XQuery 算法实现

Fig. 3　Algorithm implemented by XQuery

1.4.2　*方法可行性实验*　　此小节将识别算法运用于酒店管理系统中表明方法的可行性. 图 4 显示了在对住宿管理子系统运用算法后元关注点(MetaConcern)层次的横切关注点识别结果.

```
<result>
    <CrosscuttingConcern>
        <MetaConcern>检查剩余房间</MetaConcern>
        <Dispersity>2</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>保存日志</MetaConcern>
        <Dispersity>3</Dispersity>
    </CrosscuttingConcern>
</result>
```

图 4　元关注点层次横切关注点

Fig. 4　Crosscutting Concerns in MetaConcern Level

由图 4 可以看出，住宿管理子系统中有检查剩余房间和保存日志 2 个横切关注点，它们的扩散度分别为 2 和 3，与各个横切关注点相关联的连接点如表 2 所示. 其中保存日志是典型的系统级横切关注点，而检查剩余房间作为核心级关注点，但它降低了预定房间与入住登记两关注点的内聚度，因此是横切关注点. 为了与图 4 进行比较，图 5 显示了系统中的其它关注点及其扩散度.

表 2　横切关注点的连接点

Table 2　Joinpoint of crosscutting concerns

横切关注点	扩散度	连接点
检查剩余房间	2	预定房间、入住登记
保存日志	3	预定房间、入住登记、退房登记

1.4.3　*方法稳定性实验*　　通过实验表明在系统需求发生变更后方法所具有的稳定性.

在酒店管理系统中增加如下需求：系统中任何操作之前需要做权限检查. 本文方法对此需求变更所做的处理：首先在需求模型中变更对应的需求，然后重新映射于 XML 文件中，算法实现不需要做任何的变更. 其处理结果如图 6 所示.

1.4.4　*方法比较*　　此小节通过与国外的研究方法的比较进一步表明本文方法的优越性.

A. Sampaio 等人的研究方法[4]只能识别保存日志横切关注点，其局限性是显而易见的；而基于 Petri 网的方法[5]识别出的横切关注点包括了需求模型中所有的关注点，因此该方法因识别结果过于宽泛而缺乏实际意义.

```
<result>
    <CrosscuttingConcern>
        <MetaConcern>创建预定记录并分配房间</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>检查预定记录</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>消除预定记录</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>分配房间</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>创建入住记录和账单</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>检查入住记录和账单</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>计算账单</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>消除入住记录和账单</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>置空房间</MetaConcern>
        <Dispersity>1</Dispersity>
    </CrosscuttingConcern>
</result>
```

图 5 其它关注点及其扩散度

Fig. 5 Non-Crosscutting Concerns and Dispersity

```
<result>
    <CrosscuttingConcern>
        <MetaConcern>权限检查</MetaConcern>
        <Dispersity>3</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>检查剩余房间</MetaConcern>
        <Dispersity>2</Dispersity>
    </CrosscuttingConcern>
    <CrosscuttingConcern>
        <MetaConcern>保存日志</MetaConcern>
        <Dispersity>3</Dispersity>
    </CrosscuttingConcern>
</result>
```

图 6 需求变更后的横切关注点

Fig. 6 Crosscutting Concerns after Updating Requirements

2 结 语

以上叙述的需求阶段横切关注点识别方法，首先利用 Goal Model 对系统进行建模，然后结合横切关注点及扩散度的定义给出了横切关注点识别算法.

尽管实验结果显示此方法能识别横切关注点，但却不能识别横切关注点的类型，即不能确定横切关注点是属于 before 型，还是 after 型，甚至 around 型，导致此方法不能对需求阶段横切关注点的封装提供支持，因此未来仍需对本文方法进行更多的研究.

参考文献：

[1] 徐宝文，周超洪，周天琳，等. 面向方面的程序设计：概念、实现与未来[J]. 计算机与数字工程，2005，33(8)：1-10.

[2] 何成万，李健，焦素廷. 基于 MVC 模式的科研成果管理系统开发[J]. 武汉工程大学学报，2009，31(1)：79-82.

[3] He Chengwan，Tu Chengmao. GPRN：A Hierarchical Framework for Aspect-oriented Requirement Modeling[J]. presented in International Journal of Digital Content Technology and its Applications，February 2011，5(2)：165-172.

[4] Sampaio A，Anais Rashid. Mining Early Aspects from Requirements with EA-Miner [C]. lCSE' 08，2008：911-912.

[5] Vahdat Abdelzad，Fereidoon Shams Aliee. A method based on Petri Nets for identification of aspects[J]. presented in information science and technologies Bulletin of the ACM Slovakia，2010，2(1)：43-49.

[6] KAOS [EB/OL]. http://www. objectiver. com/fileadmin/download/documents/KaosTutorial. pdf，2011-4-10.

[7] 何成万，焦素廷，李健. 支持协同工作的加解密系统设计与实现[J]. 武汉工程大学学报，2009，31(3)：74-75.

Method based on KAOS and XML for identification of crosscutting concern

TU Cheng - mao，HE Cheng - wan

(School of Computer Science and Engineering，Wuhan Institute of Technology，Wuhan 430074，China)

Abstract：A method based on KAOS (know ledge acquisition in automated specification) and XML was proposed in this paper. First，requirement model of desired system was established by goal model of KAOS. Then the requirement model was mapped into the XML file. Last，the algorithm was presented to identify crosscutting concerns and the result of applying the algorithm to the experiment displays the feasibility and stability of the method.

Key words：identification of crosscutting concern；aspect oriented requirment engineering；knowledge acquisition in automated specification；extensible markup language

本文编辑：陈小平