基于语义技术的软件用户访问控制方法

郑高山 应 时 吴 睿
(武汉大学软件工程国家重点实验室 武汉 430072) (武汉大学计算机学院 武汉 430072)

摘 要 在应用软件中广泛使用的访问控制模型不能根据用户上下文来动态改变资源的访问权限。针对上述问题提出一种基于语义技术的访问控制方法,实现了对用户的动态授权。提出基于语义信息的用户模型和资源模型并构建面向用户模型和资源模型的基础本体,定义一组与访问控制相关的语义规则及推理规则,并设计基于语义推理过程的判定算法。访问控制过程是接收并分析访问请求,根据语义规则从显示的本体知识中获取相关联的用户信息,调用判定算法得出用户与资源间的访问权限关系。最后通过某综合减灾应用系统案例来验证该方法的有效性。

关键词 访问控制,上下文,推理规则,本体
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.8.028

Semantic-based Access Control Approach for Software User

ZHENG Gao-shan YING Shi WU Rui
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)
(Computer School, Wuhan University, Wuhan 430072, China)

Abstract The access control model widely used in the application software can't dynamically change the resource access permissions according to the user-context. This paper proposed a semantic-based access control approach which realizes the dynamic authorizing to users. Firstly, we proposed a user model and a resource model based on semantic information, built the foundation ontology for the user model and the resource model, then defined a set of semantic rules and inference rules, and designed a decision algorithm based on semantic reasoning process. The process of the approach is that receiving and analyzing access requests, obtaining the related user information from the ontology knowledge according to the semantic rules, and invoking the decision algorithm to generate the final recommended results. Finally, a case of comprehensive disaster reduction system was studied to validate the effectiveness of the approach.

Keywords Access control, Context, Inference rule, Ontology

1 引言

访问控制技术是保证信息资源不被非法访问的重要技术之一,该技术通过对用户访问资源的活动进行有效监控,使合法的用户获得有效的资源访问权限,并防止非法用户访问系统资源,从而保证系统资源的正确使用[1]。早期访问控制模型有自主访问控制模型(Discretionary Access Control, DAC)和强制访问控制模型(Mandatory Access Control, MAC)。DAC和MAC是为了解决大型主机中数据的访问权限管理问题,不能满足实际应用软件对其它资源的访问控制需求。因此,出现了基于角色的访问控制(Role-Based Access Control, RBAC)模型。

在RBAC模型中,管理员根据组织中的工作职能来创建角色,并给角色分配权限,然后将角色指派给用户,用户通过指派的角色获得相应的权限,它是一种管理企业级系统的有效技术。然而在开放、分布式的互联网环境下,随着用户上下文的改变,用户对资源的访问权限也需要相应改变。用户上下文是指用户在与应用软件交互过程中可用于表征用户状态的信息,如用户角色、部门、用户之间的关系、部门之间的关系等。基于角色的访问控制模型主要依靠指派给用户的角色设定资源的访问权限。角色与权限之间的关系是静态设定的,不能根据用户上下文动态改变用户对资源的访问权限。针对上述问题,提出了一种基于语义技术的访问控制方法,该方法可以根据用户上下文来动态改变用户对资源的访问权限,以满足开放、分布式的互联网环境下软件用户的需求。

本文提出的访问控制方法使用语义网技术来实现。语义网是由Tim Berners-Lee提出的对现有网络的扩展。网络本体语言(Web Ontology Language, OWL)是语义网的一个组

到稿日期:2015-07-29 返修日期:2015-11-21 本文受国家自然科学基金项目:面向运行性能改善的SaaS软件部署方案自优化方法(61373038),国家自然科学基金项目:基于框架的面向服务软件异常处理方法研究(61070012),国家863计划项目:网构化软件运行支撑和在线管理技术与机制(2012AA011204)资助。

郑高山(1991-),男,硕士生,主要研究方向为语义网、云安全;应 时(1965-),男,博士,教授,主要研究方向为软件工程中的智能分析与优化、软件工程的形式化理论与方法、云计算与软件服务、大数据的处理与智能分析;吴 睿(1983-),男,博士生,主要研究方向为语义网、语义搜索、规则推理。

成部分，被设计用来处理信息的内容而不仅仅是呈现信息。OWL 语言在表达能力上具有一定的局限性，规则能够弥补 OWL 在表达能力上的不足。RIF(Rule Interchange Format)规则语言是 W3C 的最新推荐标准，其致力于语义网中不同规则语言之间的规则互换。然而结合使用 OWL 与 RIF 相对复杂，语义网规则语言(Semantic Web Rule Language，SWRL)是 OWL 与 RuleML(The Rule Markup Language)的结合，能够很方便地与 OWL 结合使用，因此本文使用 SWRL 制定推理规则。

基于以上内容，本文提出了一个基于语义信息的用户模型和资源模型，利用 OWL 网络本体语言建立本体，构建本体描述用户模型和资源模型各自的概念、概念关系以及用户模型和资源模型之间的关系；在此基础上使用 SWRL 语义规则语言定义一组相关的推理规则，通过对本体进行推理生成新的语义知识，并设计了基于语义推理过程的判定算法。访问控制过程是接收用户的访问请求，根据请求内容从本体知识中获取相关的用户上下文信息，并调用判定算法得出用户和资源之间的访问权限关系。

本文第 1 节说明研究背景与研究动机；第 2 节分析国内外相关的研究情况，并与已有的工作进行对比；第 3 节详细描述了用户模型、资源模型和基础本体；第 4 节说明了基于本体的推理规则；第 5 节详细讨论了基于语义推理过程的判定算法；第 6 节通过减灾系统案例来验证所提方法的有效性；最后进行总结并说明未来的研究工作。

2 相关工作

基于角色的访问控制(RBAC)模型为访问控制领域奠定了基础，随后 Ferraiolo 对 RBAC 模型进行了多次改进，使其更加完善[2-5]。RBAC 模型解决了因自主访问控制过于灵活而带来的安全问题以及强制访问控制中主客体等级紧耦合带来的不可扩展性问题，但不能有效管理大型企业级系统中大量的用户及角色。ARBAC97 模型对 RBAC 模型进行了相应的改进，基本思想是使用 RBAC 模型对其自身进行管理，为管理层提供了方便性和可扩展性。ARBAC99 模型将用户/权限分为动态和静止两类，具有良好的可扩展性和灵活性。随后，ARBAC02 模型提出了“组织”的概念，使用户、权限与角色相对独立，改善了用户角色、角色权限冗余的状况。RBAC 模型主要依靠主体的标识来设定主体的访问权限，忽略了其它层次的信息，已经不能适应开放、分布式的互联网环境的要求。

在开放、分布式的互联网环境下，信息的交互逐步从局域网转向广域网。已有一些学者针对新型的网络环境做了相关研究。TMAC(Team-based Access Control)模型[6]以团队为核心，并用这一抽象实体来支持分布式环境中对协作用户群体安全策略的表达需求。TBAC(Task-Based Access Control)模型[7]根据用户在工作流中所执行任务为用户分配不同的资源访问权限，解决了根据用户在工作流中所执行的任务来访问资源的问题。面向服务计算的访问控制模型[8](A Service Computing oriented Access Control)通过构建服务中各个实体间的关系模型，来表示服务所属管理域之间的信任关系，根据管理域之间的信任关系来控制服务之间的相互访问。上述模型能够较好地适应网络环境中协同工作与跨域访问的特点，但授权过程都较为复杂，在授权管理上具有一定的难度。

目前互联网正向 Web3.0 发展，语义技术是 Web3.0 的重要技术，其有助于定义具有更加完好含义的信息，使信息更易于在 Web 中处理和使用，因此被运用到各个领域之中。有部分研究人员也已经将语义技术应用到访问控制领域[9-12]。以社会关系为核心的访问控制模型[9]通过构建本体形式化表达人物关系模型，将权限分为完全访问权限和受限访问权限，并根据人物之间的关系制定访问控制规则，实现了细粒度的动态访问授权，但未考虑其它层次的信息，如部门、角色等。文献[10]提出一种面向 Web 服务的访问控制方法，在 RBAC 模型中引入用户证书的概念，通过构建上层本体来描述用户证书、角色、服务及其之间的关系。根据服务的相关属性决定角色的访问权限，依据用户证书的属性来分配角色，但未能解决 RBAC 模型中静态分配角色权限的问题。文献[11]提出一个上下文感知的访问控制策略框架，在访问控制策略中考虑与用户、角色相关联的动态属性，实现了通过与角色相关联的属性对用户进行动态分配资源的访问权限。该方法的主要局限在于难以管理大量的角色及角色属性。文献[12]提出了一基于本体的上下文感知的访问控制方法，定义了一个上层用户资源描述本体，将上下文实体分为核心实体和环境实体，并制定了相关的推理规则，根据环境实体信息和推理规则得出用户与资源的访问权限关系。该方法对于核心实体的划分比较复杂，不具有良好的可扩展性。本文在上述研究的基础上深入研究了与用户有关的其它层次的信息，实现了根据用户上下文对用户进行动态授权，并将访问权限作为用户和资源间的关系，而不是实体，简化了对角色、权限及角色权限分配的管理。

3 用户模型和资源模型

文章提出了一个基于语义信息的用户模型和资源模型。在用户模型中考虑用户上下文中的相关概念和概念关系，在资源模型中考虑所需访问控制的软件资源及资源之间的关系，并通过构建本体来描述用户模型和资源模型中的概念和概念关系。

3.1 用户模型

用户模型用于形式表达用户上下文，其中包括用户、用户组、角色、部门、用户关系、部门关系。

1)用户：软件资源的使用者。

2)用户组：具有相同权限的用户集合。

3)角色：根据组织内部的工作职能划分的权限集合。

4)部门：组织中的机构。

5)用户关系：用户之间的具体关系，如上下级关系。

6)部门关系：部门之间的具体关系，如合作关系。

3.2 资源模型

资源模型形式表达所需访问控制的客体，中包括资源和资源关系。

1)资源：所有需要访问控制的客体。

2)资源关系：资源之间的具体关系，如包含关系。

3.3 基础本体

通过构建本体来描述用户模型和资源模型各自的概念、概念关系以及用户模型和资源模型之间的关系。用户模型和

资源模型之间的关系如下。

1)用户和资源之间的访问权限关系。

2)用户组和资源之间的访问权限关系。

3)部门和资源之间的所属关系。

4)角色和资源之间的访问权限关系。

出于对特定领域中的问题和具体工程需求的考虑,构建本体的过程各不相同。目前还没有一套标准的本体构建准则,其中最具有影响力的是 Gruber 在 1995 年提出的 5 条准则:1)明确性和客观性;2)完整性;3)一致性;4)可扩展性;5)最少约束。本文依据这 5 条准则构建用户模型和资源模型的基础本体。

定义 1 将基础本体定义为一个四元组(C, I, R^C, $RULE$),其中,C 表示用户模型和资源模型中所有概念的集合;V 表示 C 中概念个体的集合;R^C 表示两个概念之间的关系的集合;$RULE$ 表示基础本体中的推理规则。

定义 2(概念集) C={User, UserGroup, Role, Department, Resource},枚举了基础本体概念集中的所有元素。

定义 3(关系集) R^C={hasRole, hasDepart, hasGroup, belongTo, userRelat, departRelat, resourceRelat, canAccess},各关系的具体含义如下。

(1)hasRole。hasRole 表示用户和角色之间的所属关系。记 user={x|x 是 User 的个体},role={x|x 是 Role 的个体}。hasRole(user, role)表示用户 user 拥有角色 role。

(2)hasDepart。hasDepart 表示用户和部门之间的所属关系。记 user={x|x 是 User 的个体},department={x|x 是 Department 的个体}。hasDepart(user, department)表示用户 user 属于部门 department。

(3)hasGourp。hasGourp 表示用户和用户组之间的所属关系。记 user={x|x 是 User 的个体},userGroup={x|x 是概念 UserGroup 的个体}。hasGourp(user, userGroup)表示用户 user 属于用户组 userGroup。

(4)canAccess。canAccess 表示主体(包括用户、用户组、角色)和资源之间的访问权限关系。记 object={x|x 是 User 或 UserGroup 或者 Role 的个体},resource={x|x 是 Resource 的个体}。canAccess(object, resource)表示主体 object 能够访问资源 resource。

(5)belongTo。belongTo 表示部门和资源之间的所属关系。记 department={x|x 是 Department 的个体},resource={x|x是 Resource 的个体}。belongTo (resource, department)表示资源 resource 属于部门 department。

(6)superiorOf。superiorOf 表示用户之间的上下级关系。记 $u1$,$u2$={x|x 是 User 的个体},superiorOf($u1$,$u2$)表示用户 $u1$ 是用户 $u2$ 的上级。

(7)cooperateWith。cooperateWith 表示部门之间的合作关系。记 $d1$,$d2$={x|x 是 Department 的个体},cooperateWith($d1$,$d2$)表示部门 $d1$ 和 $d2$ 之间有合作关系。

(8)hasPart。hasPart 表示资源之间的包含关系。记 $r1$,$r2$={x|x 是 Resource 的个体},hasPart($r1$,$r2$)表示资源 $r1$ 包含资源 $r2$。

通过所构建的基础本体形式化描述用户模型和资源模型,基础本体如图 1 所示。

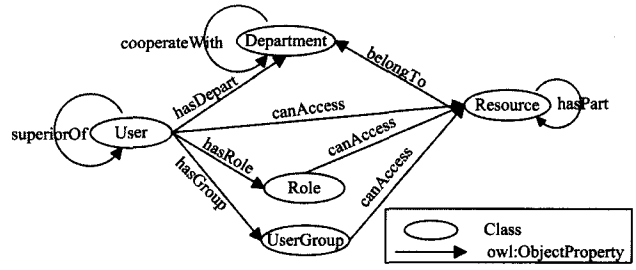

图 1 基础本体

4 推理规则

用户和资源之间的访问权限关系依赖于多个用户上下文因素,OWL 在表达关系的能力上有一定的局限性,据此研究者提出了语义网规则语言 SWRL,它是 OWL 与 RuleML 的结合,采取了一种将本体和规则相分离的策略,具有丰富的关系表达能力[13],因此通过定义推理规则来增强本体的表达能力。本文使用 SWRL 规则语言来定义与访问控制相关的推理规则。

定义 4 $RULE_1$ = User($?u$) ∧ UserGroup($?ug$) ∧ Resource($?re$) ∧ canAccess($?ug$,$?re$) ∧ hasGroup($?u$,$?ug$) → canAccess($?u$,$?re$)

$RULE_1$ 表示如果用户$?u$ 属于工作组$?ug$ 并且工作$?ug$ 可以访问资源$?re$,那么用户$?u$ 也可以访问资源$?re$。

定义 5 $RULE_2$ = User($?u$) ∧ Role($?r$) ∧ Department($?d$) ∧ Resource($?re$) ∧ hasRole($?u$,$?r$) ∧ hasDepart($?u$,$?d$) ∧ belongTo($?re$,$?d$) ∧ canAccess($?r$,$?re$) → canAccess($?u$,$?re$)

$RULE_2$ 表示如果用户$?u$ 拥有角色$?r$,角色$?r$ 可以访问资源$?re$,用户$?u$ 属于部门$?d$ 并且资源$?re$ 属于部门$?d$,则用户$?u$ 可以访问资源$?re$。

定义 6 $RULE_3$ = User($?u$) ∧ Role($?r1$) ∧ Role($?r2$) ∧ hasRole($?u$,$?r1$) ∧ subClassOf($?r1$,$?r2$) → hasRole($?u$,$?r2$)

$RULE_3$ 表示如果用户$?u$ 拥有角色$?r1$ 并且角色$?r1$ 是角色$?r2$ 的子类,那么用户$?u$ 也拥有角色$?r2$。

定义 7 $RULE_4$ = Department($?d1$) ∧ Department($?d2$) ∧ Resource($?re$) ∧ User($?u$) ∧ Role($?r$) ∧ cooperateWith($?d1$, $?d2$) ∧ belongTo($?re$,$?d1$) ∧ hasDepart($?u$,$?d2$) ∧ hasRole($?u$,$?r$) ∧ canAccess($?r$,$?re$) → hasRole($?u$,$?re$)

$RULE_4$ 表示如果用户$?u$ 属于部门$?d2$,用户$?u$ 拥有角色$?r$,角色$?r$ 可以访问资源$?re$,部门$?d1$,$?d2$ 有合作关系,资源$?re$ 属于部门$?d1$,则用户$?u$ 可以访问资源$?re$。

定义 8 $RULE_5$ = User($?u1$) ∧ User($?u2$) ∧ Resource($?re$) ∧ superiorOf($?u1$,$?u2$) ∧ canAccess($?u2$,$?re$) → hasRole($?u1$,$?re$)

$RULE_5$ 表示如果用户$?u1$ 是用户$?u2$ 的上级并且$?u2$ 可以访问资源$?re$,则用户$?u1$ 可以访问资源$?re$。

定义 9 $RULE_6$ = User($?u$) ∧ Resource($?re1$) ∧ Resource($?re2$) ∧ hasPart($?re1$,$?re2$) ∧ canAccess($?u$,$?re1$) → hasRole($?u$,$?re2$)

$RULE_6$ 表示如果用户$?u$ 可以访问资源$?re1$ 并且$?re1$ 包含资源$?re2$,则用户可以访问资源$?re2$。

上述推理规则是面向访问控制机制的基本规则,它基于本体中的实例和属性。推理机能够根据基本事实和推理规则推理得出新的语义知识,并将新的语义知识添加到本体知识

库中，从而丰富已有的知识库。

5 基于语义推理过程的判定算法

基于语义推理过程的判定算法首先分析接收到的访问请求，获取相关的用户信息与资源信息。然后根据语义规则，从显式的本体知识中获取与用户、资源相关联的其它信息，如用户角色、用户所属部门等。最后根据上述信息获取与访问权限相关的基本事实，得出用户与资源间的访问权限关系，并返回结果。访问请求分为两种：1)某个用户能够访问的所有资源；2)能够访问某一资源的所有用户。

针对上述请求设计了两种不同的判定算法。

通过集合描述法来描述基本操作及概念，判定算法中会用到相关的基本操作，判定算法 1 的基本操作定义如下。

定义 10 根据用户获取角色集 GetRoleByUser(u)

记 user=$\{x|x$ 是 User 的个体$\}$，role=$\{x|x$ 是 Role 的个体$\}$，GetRoles(u)=$\{r|$hasRole(u,r)，$u\in$ user，$r\in$ role$\}$，定义了所有和特定用户存在 hasRole 关系的角色集，其 SPARQL 实现语句如下：

SELECT ?r WHERE $\{?u$:hasRole ?$r\}$

定义 11 获取用户组 GetGroups(u)

记 user=$\{x|x$ 是 User 的个体$\}$，userGroup=$\{x|x$ 是 UserGroup 的个体$\}$，GetGroups(u)=$\{ug|$hasGroup(u,ug)，$u\in$ user，$ug\in$ userGroup$\}$，定义了所有和特定用户存在 hasGroup 关系的用户组，其 SPARQL 实现语句如下：

SELECT ?ug WHERE $\{?u$:hasGroup ?$ug\}$

定义 12 根据用户获取部门集 GetDepartByUser(u)

记 user=$\{x|x$ 是 User 的个体$\}$，department=$\{x|x$ 是 Department 的个体$\}$，GetDepart(u)=$\{d|$hasDepart(u,d)，$u\in$ user，$d\in$ department$\}$，定义了所有和特定用户存在 hasDepart 关系的部门，其 SPARQL 实现语句如下：

SELECT ?d WHERE $\{?u$:hasDepart ?$d\}$

定义 13 获取资源集 GetResources(o)

记 object=$\{x|x$ 是 User 或者 Role 或者 UserGourp 的个体$\}$，resource=$\{x|x$ 是 Resource 的个体$\}$，则GetResource(o)=$\{re|$canAccess(o,re)，$o\in$ object，$re\in$ resource$\}$，定义了所有和特定用户、角色或者用户组存在 canAccess 关系的资源，其 SPARQL 实现语句如下：

SELECT ?o WHERE $\{?o$:canAccess ?$re\}$

定义 14 根据部门获取资源集 GetResourceByDepart(d)

记 resource=$\{x|x$ 是 Resource 的个体$\}$，department=$\{x|x$ 是 Department 的个体$\}$，GetResourceByDepart(d)=$\{d|$belongTo(re,d)，$d\in$ department，$re\in$ resource$\}$，定义了所有和特定部门存在 belongTo 关系的资源，其 SPARQL 实现语句如下：

SELECT ?re WHERE $\{?re$:belongTo ?$d\}$

算法 1 某个用户能够访问的所有资源

输入：用户 user

输出：所有能访问的资源集 Resources

```
1. Groups←∅
2. Departs←∅
3. Roles←∅
4. Resources←∅
5. Gourps←getGroups(user)
6. Departs←getDepartByUser(user)
7. Roles←getRoleByUser(user)
8. for i←0 to Roles. length
9. ResourcesR←Merge(Resources,getResources(Roles[i]))
10. end for
11. for i←0 to Departs. length
12. ResourcesD←Merge(Resources,getResourcesByDepart
    (Departs[i]))
13. end for
14. for i←0 to ResourcesR. length
15. for j←0 to ResourcesD. length
16.   if ResourcesR[i]=ResourcesD[j]
17.     Resources←Merge(Resources,ResourcesR[i])
18.     break
19.   end if
20. end for
21. end for
22. for i←0 to Groups. length
23. Resources←Merge(Resources,getResources(Groups[i]))
24. end for
25. Resources←Merge(Resources,getResources(user))
26. return Resources
```

判定算法 2 会用到的基本操作定义如下。

定义 15 根据资源获取部门集 GetDepartByResource(re)

记 resource=$\{x|x$ 是 Resource 的个体$\}$，department=$\{x|x$ 是Department 的个体$\}$，GetDepartByResource(re)=$\{d|$belongTo(re,d)，$d\in$ department，$re\in$ resource$\}$，定义了所有和特定资源存在 belongTo 关系的部门，其 SPARQL 实现语句如下：

SELECT ?d WHERE $\{?re$:belongTo ?$d\}$

定义 16 根据资源获取角色集 GetRoleByResource(re)

记 role=$\{x|x$ 是 Role 的个体$\}$，resource=$\{x|x$ 是 Resource 的个体$\}$，GetRoleByResource(re)=$\{r|$canAccess(r,re)，$r\in$ role，$re\in$ resource$\}$，定义了所有和特定资源存在 canAccess 关系的角色，其 SPARQL 实现语句如下：

SELECT ?r WHERE $\{?r$:canAccess ?$re\}$

定义 17 根据资源获取用户组 GetGroupByResource(re)

记 group=$\{x|x$ 是 UserGroup 的个体$\}$，resource=$\{x|x$ 是 Resource 的个体$\}$，GetGroupByResource(re)=$\{r|$canAccess(ug,re)，$ug\in$ group，$re\in$ resource$\}$，定义了所有和特定资源存在 canAccess 关系的用户组，其 SPARQL 实现语句如下：

SELECT ?ug WHERE $\{?ug$:canAccess ?$re\}$

定义 18 根据角色获取用户集 GetUserByRole(r)

记 user=$\{x|x$ 是 User 的个体$\}$，role=$\{x|x$ 是 Role 的个体$\}$，GetUserByRole(r)=$\{u|$hasRole(u,r)，$u\in$ user，$r\in$ role$\}$，定义了所有和特定角色存在 hasRole 关系的用户，其 SPARQL 实现语句如下：

SELECT ?u WHERE $\{?u$:hasRole ?$r\}$

定义 19 根据部门获取用户集 GetUserByDepart(d)

记 user=$\{x|x$ 是 User 的个体$\}$，department=$\{x|x$ 是 Department 的个体$\}$，GetUserByDepart(d)=$\{u|$hasDepart(u,d)，$u\in$ user，$d\in$ department$\}$，定义了所有和特定部门存在 hasDepart 关系的用户，其 SPARQL 实现语句如下：

SELECT ?u WHERE $\{?u$:hasDepart ?$d\}$

定义 20 根据用户组获取用户集 GetUserByGroup(ug)

记 user=$\{x|x$ 是 User 的个体$\}$，userGroup=$\{x|x$ 是

UserGroup 的个体}，GetUserByResource(ug)＝{u|hasGroup(u,ug)，u∈user，ug∈userGroup}，定义了所有和特定用户组存在 hasGroup 关系的用户，其 SPARQL 实现语句如下：

SELECT ?u WHERE {?u:hasGroup ?ug}

定义 21 根据资源获取用户集 GetUserByResource(re)

记 user＝{x|x 是 User 的个体}，resource＝{x|x 是 Resource 的个体}，GetUserByResource(re)＝{u|canAccess(u,re)，u∈user，re∈resource}，定义了所有和特定资源存在 canAccess 关系的用户，其 SPARQL 实现语句如下：

SELECT ?u WHERE {?u :canAccess ?re}

算法 2 能够访问某一资源的所有用户

```
输入：资源 resource
输出：所有能访问该资源的用户集 Users
1. Groups←∅
2. Departs←∅
3. Roles←∅
4. Resources←∅
5. Gourps←getGroupsByResource(resource)
6. Departs←getDepartByResource(resource)
7. Roles←getRoleByResource (resource)
8. for i←0 to Roles. length
9. UsersR←Merge(UserR,getUserByRole(Roles[i]))
10. end for
11. for i←0 to Departs. length
12. UsersD←Merge(UserR,getUserByDepart(Departs[i]))
13. end for
14. for i←0 to UsersR. length
15.    for j←0 to UsersD. length
16.       if UsersR[i]=UsersD[j]
17.          Users←Merge(Users,UsersR[i])
18.          break
19.       end if
20.    end for
21. end for
22. for i←0 to Groups. length
23. Users←Merge(Users,getUserByGroup(Groups[i]))
24. end for
25. Users←Merge(Users,getUserByResource(resource))
26. return Users
```

算法的复杂度跟各概念的个体数有关，假设资源的个体数为 m，角色的个体数为 r，部门的个体数为 d，用户的个体数为 u，工作组的个体数为 g。设 n＝max(r,d,g)，则算法 1 的时间复杂度近似为 O(mn)，算法 2 的时间复杂度近似为 O(un)。

6 案例应用

本案例来自某综合减灾空间信息服务应用系统，目标是提供灾害分析服务，作出灾害应急响应决策。未发生灾害时，各部门内部人员根据自身指派的角色访问指定的数据资源，保证日常工作顺利完成，相互不会影响。在灾害发生时，各部门需要指派人员相互协作，通过对各部门的各类数据进行综合分析处理得出准确的综合报告，及时发布权威信息以便最大程度地减少灾害地区的损失。本文所提出的访问控制方法成功应用在了这一项目中并在一定程度上解决了上述问题，由于实际情况十分复杂，本文只以相关重点部分作为案例进行介绍。

1)未发生灾害时，卫星遥感部、航空遥感部的各人员根据自身指派的角色分析处理所指定的各类数据，如卫星原始数据、航空原始数据、卫星预评估数据、航空预评估数据等。

2)灾害发生时，数据源中会新增灾害现场数据，各部门需要相互协作，并指派人员参与到应急用户组中，通过对各部门的各类数据进行综合分析处理得出准确的综合报告。

6.1 领域本体

领域本体根据具体工程需求对基础本体进行扩展，本案例所扩展的领域本体如下。

角色 Role 的子类有：Director(部门主任)、Analyst(分析师)、Evaluator(评估员)。分析师负责分析原始数据，生成预评估数据；评估员负责处理预评估数据，生成评估数据；部门主任负责审核评估数据。

资源 Resource 的子类有：CompreReport(综合报告)、EvaluateDate(评估数据)、AviEvaData(航空评估数据)、SatEvaData(卫星评估数据)、PreEvaData(预评估数据)、AviPreData(航空预评估数据)、SatPreData(卫星预评估数据)、InitialData(原始数据)、AviIniData(航空原始数据)、SatIniData(卫星原始数据)、FieldData(现场数据)。

部门 Department 的子类有：SatDepart(卫星遥感部)、AviDepart(航空遥感部)。

扩展的领域本体如图 2 所示。

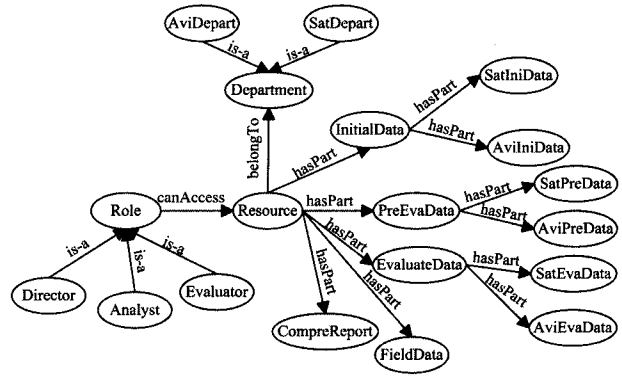

图 2 部分领域本体

6.2 领域个体

个体表示某个概念的实例，根据减灾项目对访问控制的具体需求建立个体，案例的知识库由个体及其关系组成。

User＝{U_1,U_2,U_3,U_4,U_5,U_6}

Department＝{D_{sat}(卫星遥感部)，D_{avi}(航空遥感部)}

Role＝{R_{dir}(部门主任)，R_{ana}(分析师)，R_{eva}(评估员)}

UserGroup＝{UG_{Eme}(应急用户组)}

Resource＝{Re_{CPR}(综合报告)，Re_{ED}(评估数据)，Re_{AED}(航空评估数据)，Re_{SED}(卫星评估数据)，Re_{PED}(预评估数据)，Re_{APD}(航空预评估数据)，Re_{SPD}(卫星预评估数据)，Re_{ID}(原始数据)，Re_{FD}(现场数据)，Re_{AID}(航空原始数据)，Re_{SID}(卫星原始数据)}

个体间关系如下：

hasDepart＝{(U_1,D_{sat})，(U_2,D_{sat})，(U_3,D_{sat})，(U_4,D_{avi})，(U_5,D_{avi})，(U_6,D_{avi})}

hasRole＝{(U_1,R_{dir})，(U_2,R_{ana})，(U_3,R_{eva})，(U_4,R_{dir})，(U_5,R_{ana})，(U_6,R_{eva})}

canAccess＝{(R_{dir},Re_{ED})，(R_{ana},Re_{ID})，(R_{eva},Re_{PED})，(UG_{Eme},Re_{FD})，(UG_{Eme},Re_{CPR})}

belongTo＝{(Re_{SED},D_{sat})，(Re_{SPD},D_{sat})，(Re_{SID},D_{sat})，

$(Re_{AED}, D_{avi}), (Re_{APD}, D_{avi}), (Re_{AID}, D_{avi})\}$

其中部分个体的 OWL 描述如下：

```
〈User rdf:ID="U1"〉
    〈hasRole rdf:resource="#Rdir"/〉
    〈hasDepart rdf:resource="#Dsat"/〉
〈/User〉
〈Role rdf:ID="Rdir"〉
    〈canAccess rdf:resource="#ReED"/〉
〈/Role〉
〈Resource rdf:ID="ReED"〉
    〈belongTo rdf:resource="#Dsat"/〉
〈/Resource〉
〈Department rdf:ID="Dsat"/〉
...
```

6.3 实验验证

在现有知识库的基础上，通过分析灾害发生前后用户与资源间的访问权限关系来验证方法的有效性。

在未发生灾害时，卫星遥感部和航空遥感部的各用户访问部门内部的相关资源，完成部门内部的日常工作。

当灾难发生时，各部门需要相互协作，部门 Dsat 和 Davi 建立合作关系：

cooperateWith=$\{(D_{sat}, D_{avi})\}$

添加应急用户组成员，用户 U_1，U_3，U_6 添加到应急用户组中；

hasGroup=$\{(U_1, UG_{Eme}), (U_3, UG_{Eme}), (U_6, UG_{Eme})\}$

根据用户上下文的改变及推理过程得出当前用户与资源间的访问权限关系，实验结果如表 1 所列。

表 1 实验结果

个体关系 / 用户个体	未发生灾害				灾害发生后	
User	hasDepartment	hasRole	hasGroup	canAccess	hasGroup	canAccess
U_1	Dsat	Rdir		ReSED	UGEme	ReSED, ReAED, ReCPD
U_2	Dsat	Rana		ReSID		ReSID, ReAID
U_3	Dsat	Reva		ReSPD	UGEme	ReSPD, ReAPD, ReFD
U_4	Davi	Rdir		ReAED		ReSED, ReAED
U_5	Davi	Rana		ReAID		ReSID, ReAID
U_6	Davi	Reva		ReAPD		ReSPD, ReAPD

从表 1 中的实验结果可知，在未发生灾害时，卫星遥感部和航空遥感部的各用户访问部门内部的相关资源，完成部门内部的日常工作。在灾害发生后，U_3 可以访问 Re_{APD}，U_6 可以访问 Re_{SPD}，并且 U_3，U_6 可以访问 Re_{FD}，因此，U_3，U_6 可以综合分析 Re_{APD}，Re_{SPD} 和 Re_{FD}，生成 Re_{CPR}（综合报告）。U_1 可以访问 Re_{CPR}，能够对 Re_{CPR} 进行审核。实验结果表明该方法可以根据用户上下文来动态改变用户对资源的访问权限，以满足开放、分布式的互联网环境下软件用户的需求。

结束语 本文提出一种基于语义技术的访问控制方法，定义了面向用户模型和资源模型的基础本体并构建了基本推理规则，详细讨论了确定用户和资源之间访问权限关系的判定算法，实现了根据用户上下文对用户进行动态授权。并通过某综合减灾应用系统案例来验证该方法的有效性，实验结果表明该方法支持根据用户上下文动态改变用户对资源的访问权限。该方法具有较高的可扩展性，应该能够适应云环境下软件对访问控制机制的需求，后续将对其进行进一步研究。

参 考 文 献

[1] Li Feng-hua, Su Mang, Shi Guo-zhen, et al. R-esearch Status and Development Trends of Access C-ontrol Model[J]. Acta Electronica Sinica, 2012, 40(4): 805-813(in Chinese)
李凤华，苏锦，史国振，等. 访问控制模型研究进展及发展趋势[J]. 电子学报，2012，40(4)：805-813

[2] Sandhu R S, Coyne E J. Role-based access control models[J]. Computer, 1996, 29(2): 38-47

[3] Sandhu R, Bhamidipati V, Munawer Q. The ARBAC97 model for role-based administration of roles[J]. Acm Transactions on Information & System Security, 1999, 2(1): 105-135

[4] Munawer Q, Sandhu R. The ARBAC99 Model for Administration of Roles[C] // Computer Security Applications Conference, Annual. IEEE Computer Society, 1999: 229-238

[5] Oh S, Sandhu R. A model for role administration using organization structure[C] // Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies. ACM, 2002: 155-162

[6] Alotaiby F T, Chen J X. A Model for Team-based Access Control[C] // International Conference on Information Technology: Coding & Computing. IEEE Computer Society, 2004: 450-454

[7] Periorellis P, Parastatidis S. Task-Based Access Control for Virtual Organizations[M] // Scientific Engineering of Distributed Java Applications. Springer Berlin Heidelberg, 2005: 38-47

[8] Cao Chun, Ma Xiao-xing, Lv Jian. SCoAC: A Service Computing Oriented Access Control Model[J]. Chinese Journal of Computers, 2006, 29(7): 1209-1216(in Chinese)
曹春，马晓星，吕建. SCoAC：一个面向服务计算的访问控制模[J]. 计算机学报，2006，29(7)：1209-1216

[9] Chowdhury M M R, Noll J. A social relation aware semantic access control[C] // 12th International Conference on Computers and Information Technology, 2009 (ICCIT 039; 09). IEEE, 2009: 139-144

[10] He Z, Wu L, Li H, et al. Semantics-based Access Control Approach for Web Service[J]. Journal of Computers, 2011, 6(6): 1152-1161

[11] Kayes A S M, Han J, et al. A Semantic Policy Framework for Context-Aware Access Control Applications[C] // 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2013: 753-762

[12] Kayes A S M, Han J, Colman A. An Ontology-Based Approach to Context-Aware Access Control for Software Services[C] // The International Conference on Web Information System Engineering(WISE). 2013: 410-420

[13] World Wide Web Consortium. SWRL: A Semantic Web Rule Lang-uage Combining OWL and RuleML[EB/OL]. http://www.w3.org/Submission/SWRL