信息和系统安全管理策略工程研究

摘 要

策略一直在复杂系统的管理中扮演着一个重要的角色，可以用来指导企业的信息和系统的管理。基于策略的管理方法是近年来被广泛认可的、管理复杂系统的一个有效解决方案，其核心思想是使用策略来驱动管理过程，把管理策略和管理系统分离开来，根据环境变化的需要而动态改变管理策略，从而相应地改变系统的行为，无需重新实现系统，提高了管理系统的可扩展性和灵活性，增加了系统管理的实时性和自动化程度。基于策略的管理最初应用于网络的 QoS 管理领域，而在信息和系统的安全管理领域的应用也颇具吸引力，目前正吸引着越来越多的学者和研究人员的兴趣。

基于策略的管理突出了策略在管理中的作用，然而在一个复杂的信息系统中，管理策略本身就构成了一个特殊的、复杂的、相对独立的子系统，并形成于一个复杂而又完整的过程，这个过程将包括策略的定义、分析、翻译转换、发布、执行和维护等等。因此，需要在一个系统方法学的指导下， 将管理过程中的各个环节和各种任务有机地组织起来。策略生命周期的研究目的着眼于研究管理策略自身的管理，可用于指导整个基于策略的管理过程，将管理过程中的各个环节、各种管理活动和资源有机地组织起来，以确保管理策略的完整性、完全性、正确性、可实施和可维护性。作为一项重要的、原创性的研究成果，本文给出了一个合理的、完整的策略生命周期模型，明确定义了策略演化过程中各阶段的目标、任务、活动、输入/输出和参与者，以及各阶段之间的依赖关系等等。

信息和系统管理的主要目标是保证对企业信息资源的正确、高效和安全的利用，以支持企业的业务目标。编写和定义明确的、全面的、精确和完整的系统管理策略对于成功的企业信息资源的管理是至关重要的。而信息系统的管理需求分析正是编写和定义管理策略的依据和基础，在成功的系统管理中起着决定性的作用，同时也是目前该领域最薄弱环节，尚未见有正式的研究成果。 本文通过对软件工程中各种系统需求分析技术的研究和比较，采纳了基于目标的需求分析技术，在该方法的基础上，进行了相应的改进和扩展，使之能够很好地应用于信息

资源管理需求分析和管理策略的定义。基于目标的需求分析是以目标为主线，通过对目标的鉴别、定义、分析、组织和精化等来获取系统需求的一种方法。本文提供了基于目标的企业信息和系统管理需求分析框架模型，介绍了建立目标模型的步骤和方法，以及根据目标模型定义信息和系统管理策略的规则。目标障碍分析和处理也是基于目标的需求分析中的一项重要任务，本文同样讨论了目标障碍的识别和处理，除采用传统的障碍处理方法外，还提供了直接根据障碍来定义相应的管理策略的方法。另外本文还重点讨论了在系统安全管理中，如何根据恶意目标建立反目标模型，以及如何根据反目标模型确定安全需求和定义安全策略等的方法。

除了基于目标的需求分析外，还可以借鉴其它的系统需求分析方法，本节将简单介绍如何采用用况和误用用况来进行信息和系统的管理需求分析以及指导管理策略的制定。误用用况是一种特殊类型的用况，定义了不希望系统/实体发生的行为，可用来对系统中的错误和恶意行为进行建模。本文提供了建立用况和误用用况模型的方法，提供了用况和误用用况模板，以及根据用况和误用用况确定管理策略的方法。

策略的分析和验证也是策略演化过程中的另外一个重要任务。在策略演化过程中的各个阶段，都需要对相应层次的策略进行分析和验证。由于在策略的演化过程中各个阶段所处的上下文环境不同，策略的抽象层次不同，因此，各个阶段的分析和验证的侧重点和任务也不一样，所采用的分析方法和建模工具也不尽相同。本文分别在策略的高层、中间层和低层三个不同抽象层中各选出一到两项重要的、目前该领域中尚未成熟、亟待解决的一些策略分析和验证内容进行讨论。对于高层策略，本文分析了在实例化策略时应考虑的问题，例如如何反映策略之间的依赖关系、如何实现策略实例间的安全约束和控制原则、以及如何进行合理的资源分配和规划等等。本文采用彩色 Petri 网建立业务过程的授权和职责模型，利用线性代数技术分析授权状态的可达性，从而验证策略实例是否违反了安全约束和控制原则；本文还利用 Petri 网的覆盖图来计算业务过程的有效执行链，从而实现了对人力资源的合理规划。对于中间层策略，本文重点讨论了如何分析和验证中间层策略是否满足或符合高层策略的要求，是否与高层策略相矛盾等。本文以网络的访问控制策略为例，仍采用彩色 Petri 网作为主要建模和分析工具，通过提供一个经过扩展的彩色 Petri 网框架，在系统和抽象设备层上对网络、网络访问控制策略和网关的路由或过滤规则进行建模，利用 Petri 网的覆盖图验证网关配置的正确性，并且给出了如何发现和纠正错误的方法。对于低层策略，本文利用空

间几何技术对防火墙的过滤规则进行建模，帮助系统管理人员发现和消解过滤规则冲突，并能够对防火墙规则库进行彻底重写和全面优化，大大减少规则的数量，从而提高了防火墙的性能。在空间几何模型中，每个过滤规则被映射成多维空间中的一个规则几何体，从而使防火墙配置的语义可视化，对过滤规则的验证变得简单而直观。

策略的翻译、发布和实施同样是基于策略的管理中的关键环节，也是目前大多数研究人员的研究重点，本文将对它们做一些简单的探讨，简单介绍了目前策略的翻译、发布和实施中常用的一些基本方法。

综上所述，本文的主要创新性研究工作可以概括为：

- 系统地研究了信息和系统的管理策略的生命周期，首次明确提出了策略工程和策略生命周期模型的概念，并给出了一个合理的、完整的策略生命周期模型，可用以指导实施基于策略的管理方法在企业的信息和信息系统管理中的应用；
- 在策略工程概念的指导下，明确了管理需求分析在策略生命周期中的重要性，并在借鉴了已有的需求分析技术基础上，提供了基于目标的信息和系统管理需求分析技术，以及根据需求确定管理目标和管理策略的方法；
- 对于策略的分析和验证，提供了一些新的建模和分析方法，例如采用彩色 Petri 网对业务过程的授权和职责进行建模，不仅能够包含权限和职责管理，还能够包含安全约束如 SoD(Separation of Duty)以及角色层次关系结构(Role Hierarchy)等概念。对于网络访问控制，首次采用彩色 Petri 网对网络和相关访问控制策略进行建模，给出了如何根据所要进行的分析类型确定 Petri 网中弧方向的方法。在防火墙过滤规则的分析中，将原有的两个规则之间的冲突概念扩展到多个规则之间，并给出了规则库的彻底重写和全面优化的方法。

关键词： 策略工程，策略生命周期模型，信息和系统管理，基于目标的需求分析，策略验证，策略优化

STUDY ON POLICY ENGINEERING FOR SECURITY MANAGEMENT OF ENTERPRISE INFORMATION AND SYSTEMS

ABSTRACT

Policies always play an important role in the management of complex systems. It is also well suited to be used in the management of enterprise information and systems. Policy-based management (PBM) is accepted as a promising solution for management of complex systems in recent years. The main idea in PMB is the use of policies as a means of driving management procedures. In PBM, policies are separated from the management systems. Polices are changing according the needs of the changing environment and thus change the behaviors of the systems without having to re-implement the systems. This characteristic is able to improve the scalability and flexibility of the management system. The management procedure can thus be largely automated and adjusted in real-time. Policy-based management is originally applied in the area of network QoS management. Now its application in the security management of information and systems is becoming more and more attracting to the researchers in this area.

PBM emphasizes the importance of policies in the management activity. However, in a complex information system, policies themselves would constitute a special complex and relatively separated sub-system and are shaped through a complex and integrated process. This process would contain the definition, analysis, translation, distribution, enforcement and maintenance of the management policies. Therefore, all activities in the management process must be organized and performed under the guide of a systematic methodology. The purpose of the research of policy lifecycle is to study the management of policies themselves and to guide the whole policy-based management procedure so as to guarantee the integrity, completeness, correctness, enforceability and maintainability of management policies. This

dissertation provides a workable and complete policy lifecycle model and defines clearly the objectives, tasks, activities, input/output, and participants in each phase of the policy evolution process. The relationships between these phases are also given.

The main objective of the management of the enterprise information and systems is to ensure the efficient, correct and security use of its information resources and thus to support its business objectives. Clearly, completely, and precisely defined management polices are critical to the successful management of an enterprise's information resources. The requirements analysis of information and systems management is the basis of policy definition and is the decisive factor in the successful management. However, this problem has not received enough attention in the literature and no formal research result can be found currently. By comparing various requirements analysis technologies in the software engineering, we adopted the goal-based requirements analysis technology. By necessary adaptation and extension, the goal-based requirements analysis method can be applied to the analyzing of information and systems management requirements and determination of management polices quite suitably. Goal-based requirements analysis uses goals as the main clue to acquire system requirements through goal identification, analysis, definition, refinement, and operationaliztion. This dissertation provides a goal-based requirements analysis framework for management of enterprise information and systems. The process and method for establishing the goal model and the rules for defining management policies from the goal model are also provided. Another important part in goal-based requirements analysis is the analysis of obstacles to goals. This dissertation also introduces the method for the identification and handling of obstacles. Besides the traditional methods for obstacle handling, this dissertation provides a method to handle obstacles by defining management policies directly from them. This dissertation also introduces how to establish anti-goal model from malicious goals to acquire requirements in security management of information and systems, and how to derive security management policies from those anti-goals and requirements.

Besides goal-based requirements analysis, other requirements analysis techniques can also be adopted to acquire the requirements of information and systems

management. This dissertation briefly introduces the adoption of use and misuse cases for this purpose. Misuse case is a special type of use case. It defines the undesirable behaviors of the system or the entity, which are not expected to happen. Misuse case can be used to model the errors or malicious behaviors in the systems. This dissertation provides the method for the construction of use and misuse cases models and provides the definition of their templates. Management policies can thus be derived from the use and misuse cases already defined.

The analyzing and validating of management policies is another important task within the policy evolution process. The focus and contents of the analysis vary at each phase because of their different context and abstraction levels. As a result, the analysis techniques and modeling tools also vary. This paper selects one or two important and immature issues in policy analysis and validation at the high, medium, and low abstraction levels separately. For high-level policies, this paper discusses the issues that should be paid attention to during policy instantiations, such as how to reflect the dependency relationships between policies, how to implement the security constraints and control principles between policy instances, and how to allocate related resources reasonably, etc. To address these issues, this dissertation uses colored Petri net to model the authorization and obligation of business process. Algebra technique is adopted to analyze the reachable authorization states of the model to verify if any policy instance has violated the security constraints and control principles. This dissertation also uses the coverability graph of Petri net to calculate the valid execution chains of the business process, which is able to help to perform reasonable human resource allocation for business process instances. For intermediate level policies, this dissertation shows how to check if the intermediate policies comply with and satisfy the requirements of the high level policies. Colored Petri net is also used here as the modeling and analyzing tool to analyze the network access control policies. An extended colored Petri net framework is provided to model the network, network access policies and gateway configurations at the system and abstraction device level. We show how to use the coverability graph of the colored Petri net to validate the correctness of the gateway configurations at the global viewpoint of the whole network, and how to find and correct the errors in gateway configuration. For low-level policies, we present a technique by using the

geometry technology to model the firewall filtering rules. This technique can help us to detect and resolve conflicts among filtering rules, and is able to rewrite and optimize the filtering rule base completely. The number of filtering rules is largely reduced after the optimization and as a result, the performance of the firewall can be improved a lot. In the geometry model, each filtering rule is mapped onto a rule object in the multiple-dimensional hyperspace. The semantic of the firewall configuration is therefore visual to the analyst and the validation of filtering rules against the network access policies becomes intuitive and simple.

Policy translation, distribution, and enforcement are also critical phases in the policy evolution process. They are among the research interests of most researchers currently. This dissertation briefly introduces the main methods that are currently adopted in the literature.

In one word, the main original works in this dissertation are summarized below:

- Systematically study the policy lifecycle for the management of enterprise information and systems. Introduce the concept of policy engineering and policy lifecycle model for the first time in the literature and provide a complete and workable policy lifecycle model that can be used to guide the application of policy-based management in the management of enterprise information and systems;
- Clearly emphasize the importance of the management requirements analysis in the policy lifecycle under the guidance of the concept of policy engineering. Provide the goal-based requirements analysis technique for information and systems management after comparison of existing requirements analysis techniques in the software engineering, and provide the method for determine the management objectives and policies based on the acquired requirements;
- For policy analysis and validation, provide some new modeling and analyzing techniques. For example, the use of colored Petri net to model the authorization and obligation of the business process is able to contain not only the management of authorization and obligation, and also the security constraints such as separation of duties and the concept of role hierarchy. For analysis of network access control policies, the colored Petri net is used for

the first time in the literature to model the network and related access policies, and provide the method for determining the arc directions according to what type of analysis is to be performed. For analysis of firewall filtering rules, extend the concept of rule conflict between two rules to that among multiple rules and provide the method for completely rewriting and optimizing the rule base.

KEY WORDS: policy engineering, policy lifecycle model, information and systems management, goal-based requirements analysis, policy validation, policy optimization

本文表格索引

本文图索引

上海交通大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：张翼

日期：2006 年 06 月 06 日

上海交通大学
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

保密□，在____年解密后适用本授权书。

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：张骥

日期：2006 年 06 月 06 日

指导教师签名：汪石农

日期：2006 年 06 月 06 日

第一章 引言

在今天全球一体化的趋势下，企业和组织，无论大小，大都采用或依赖信息技术来创造新的商业良机，开发和提供新的产品和服务，以提高竞争力。同时，信息技术也改变了它们的生产运作方式、业务流程、甚至组织结构。随着计算机和网络技术的迅速发展，互联的计算机网络成为企业首选的 IT 基础设施。今天，一个典型的企业网络系统将包括数量众多的、分布的、异构的和互联的设备，如路由器、交换机、防火墙、服务器、工作站和 PC 等等，在这些系统上运行着各种各样的系统和应用程序，为形形色色的用户提供各种不同的服务。为了保证信息资源的可用性和安全性，提高服务质量，对企业信息资源的有效管理是十分必要的。信息资源管理的对象涉及企业所有的数据、文件、设备、服务、应用、用户，以及它（他）们之间的关系和交互。被管理系统的复杂性直接导致了系统管理的复杂性和高成本。“在一个实际的应用系统的推广和应用中，最主要的成本往往不是硬件和软件成本，它们与制定和实施系统的具体使用策略和过程，以及培训用户的“软”成本相比，往往相形见绌。这些“软”成本一般占总成本的 75%以上” [Merrill2001]。

此外，由于信息技术具有专业性、技术性强的特点，一个企业的信息系统管理人员往往仅仅由缺乏充分了解企业业务和组织结构的 IT 技术人员构成，因而很容易造成信息系统的管理与开发应用信息系统的最终目标脱节。众所周知，信息系统是为整个企业的业务服务的，信息系统的管理不能脱离系统所运行的环境，而必须是一个由企业业务驱动(Business-Driven)的，自适应性的体系。另外，在目前的技术和商业环境下，由于业务需要，如商家之间的合作关系，人员的流动，新业务和新技术的不断涌现等等，企业的信息系统的组成、使用者和服务对象是动态变化的，这就需要进行实时和动态的配置和管理，从而使得信息系统的管理更加复杂化。此时，传统的、基于管理员的知识和经验的管理方式是不能够满足需要的，其所花费的精力和时间随着系统规模的扩大呈指数级增长，管理错误和漏洞的概率也会大幅增加。另外，随着新技术新设备的采用，管理人员必须随时接受培训，及时掌握新技术新应用，或者雇佣新的管理员，这样往往增加的管理成本超过了新技术所带来的效益，使得新技术难以被推广。

目前一个业界的共识是信息和系统的管理远远落后于信息技术的发展，这一点在信息的安全管理方面尤为突出[Konstantinous1999]。信息和系统管理的研究和技术开发是目前信息技术领域一个重要而迫切的任务。

1．1 信息和系统管理的新要求

信息和系统管理的复杂性所带来的高管理成本和过长的管理周期，对信息和系统的管理提出了以下三点要求：

1． 业务驱动(Business-driven Management)

企业信息资源的管理必须以实现企业的业务为最终目标，充分体现系统运行的环境对系统管理的要求。企业的业务目标、组织原则和业务规则是信息和系统管理必须要考虑的因素，是管理工作的依据和出发点[Strassner2002a&b]。

2． 集中和分布相结合(Combination of Central and distributed management)

集中管理是为了从全局的角度合理配置资源、协调各子系统之间的合作和交互。一个企业的业务流程往往是跨部门，跨单位的，通常一个业务目标由多个系统合作完成，一个系统也可以支持多个业务目标，因而相应的信息和信息系统也因为业务流程中各个任务之间的各种依赖关系而是相互依赖和合作的。一个子系统往往因为它所依赖的其它子系统的管理漏洞而不能有效地工作。因而，企业的信息系统的管理必须考虑到各个子系统的依赖关系，确保各个子系统的协调，这一点在信息系统的安全管理中尤为重要。

信息和系统的管理同时又必须是可分布的。由于一个企业的信息系统的规模、复杂性和动态性，加上各个子系统在功能上的相对独立性和地理位置上的分散性（往往可以跨部门、跨省市甚至国界），决定了单纯的集中式的管理是低效的，因而系统的管理又必须是可分布的，各个子系统又具有一定的独立性，可以自行选择合适的方式和机制。

3． 管理过程的系统化和自动化(Systematic and Automated Management)

管理过程必须系统化和尽量自动化，避免管理中的随意性和不确定性，减少人为因素，从而降低对管理人员的知识和经验的依赖，减少管理成本和降低出错的概率，降低手工操作所带来的延迟，提高系统管理的自适应性。

基于策略的系统管理方法（Policy－Based Management）是目前越来越被广泛认可的，管理复杂系统的一个新颖、有效的方法，能够很好地实现上述对企业信息和系统管理新的要求。

1.2 策略和基于策略的管理

策略一直在复杂系统的管理中扮演着一个重要的角色,被用来指导系统和网络的管理，策略将管理活动、管理工具、被管理资源以及相关人员贯穿在一起。对于策略的定义有多种，主要是由于所面向的管理任务的不同和观察问题的角度不同而导致的。IETF 将策略定义为从广义上说“策略是一个明确的目标、或行为过程的规定和方法，用以指导现在和将来的决策”，或从狭义上说“策略是一个规则集，用以管理和控制对网络资源的访问”[RFC31982001]。本文采用 Sloman 等人的定义[Sloman1994,Damianou2002]：

策略是从管理目标得出的，持续性和说明性的规则，定义了系统的行为准则。

策略的持续性指出策略相对于被管理系统的状态来说是相对静态的，一次性的命令或指令不是策略；说明性指的是策略仅需要定义什么是理想的、可接受的行为，而无需考虑如何去实现或维持它。该定义强调了策略是从管理目标中得出的，这里管理目标应包括企业的业务目标，服务标准和协议等等。

基于策略的管理方法是近年来被广泛认可的复杂系统管理的一个有效解决方案，其核心思想是使用策略来驱动管理过程，把管理策略和管理系统分离开来，能够根据环境变化的需要而动态改变管理策略，从而可以相应地改变系统的行为，而无需重新实现系统，基于策略的管理提高了管理系统的可扩展性和灵活性，减轻了管理员的负担，并增加了系统管理的实时性和自动化程度[Damianou2002]。该方法最初应用于网络的 QoS 管理领域，现在它被证明同样可以很好地应用于其它管理领域。

在基于策略的管理中，管理者可以采用高层抽象的策略来定义系统的理想行为和状态，这些高层抽象策略逐步被翻译成可被设备理解和直接执行的低层策略，并最终得以执行[Heiler et1996]。

1.2.1 策略的层次结构

基于策略的管理对策略有以下几点要求[Verma2001]：

- 精确性。为使系统能够理解和实施策略，策略应该是精确详细和无二义的；
- 一致性。策略应该是一致的，策略之间没有冲突。
- 兼容性。策略和设备的能力是兼容的，即策略是可被系统实施的；

- 易于说明性。为了便于定义完整一致的策略，策略必须简单明了，易于说明和定义；
- 直观性。管理人员能够使用所熟悉的概念和形式来定义策略。

很明显以上这些要求是无法同时满足的，例如，为方便管理人员定义策略而要求的直观性和易于说明性，与为满足设备实施策略所要求的精确性是相互矛盾的。因此，必须根据策略的抽象层次对策略进行分类。管理员可以使用高层抽象的策略来表达他们的目标和要求，这些高层策略被翻译成底层策略，从而被设备理解和实施。一些学者主张按策略的抽象层次将策略分为四层[Verma2001]，如图 1－1 所示。

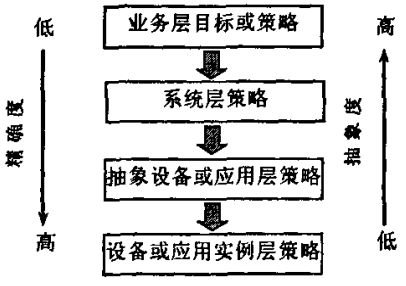


图 1－1 策略层次结构

Fig. 1-1 Policy Hierachy

- 业务层策略。这些策略是从企业的业务目标直接得出的，为实施它们，必须将它们精化成下面三层策略；
- 系统层策略。这些策略在系统功能层上实现了业务层策略，指导系统和活动的管理，规定了系统和服务的行为；
- 抽象设备或应用层策略。这些策略规定了系统中各种组成部件的行为；
- 设备或应用实例层策略。这些策略面向具体的系统设备和应用实例，可由设备直接理解和实施，与设备或应用的配置策略或文件相对应。

在实际应用中，层与层之间的界限往往是模糊不清的，根据实际需要也可以划分为三层或五层策略。高层策略需要精化和翻译成低层策略，才能由设备和人员来执行。策略翻译过程是一个逐步求精的过程，其中有的阶段是可以自动实现的，有的则必须是完全手工完成的，通常情况下，策略的翻译是计算机辅助完成的。在翻译过程中，必须能够提供足够的系统信息，如网络的配置和拓扑、设备和服务的有关信息、角色和用户信息、应用环境的信息等等。

1．2．2 基于策略的管理体系架构

IETF 提出了一个通用的基于策略的管理体系结构[IETF2001a]，可作为基于策略的管理的参考模型，如图 1—2 所示。

该框架包括策略管理工具、策略库、策略实施点和策略决策点四个组成部分。

策略管理工具是管理员用来输入、分析、验证、精化和翻译管理策略的控制台；策略实施点 PEP(Policy Enforcement Point)是能够实施策略的设备；策略库用于存储由策略控制台输入的策略；策略决策点 PDP(Policy Decision Point)处于策略实施点和策略库之间，负责查询和解释策略库中的策略，并负责将合适的策略传送给策略实施点。PDP 和 PEP 可以处于同一台设备上，也可以处于不同的设备上。这四个组成部分之间可采用不同的协议进行通讯，如 PDP 和 PEP 之间采用 COPS 或 SNMP 协议，而策略库使用 LDAP 作为访问协议，而策略的发布可以采用 PUSH 和 PULL 两种方式。

IETF 的研究重点主要在信息模型和网络的 QoS 管理，目前已有初级产品问世，而对网络和系统的安全管理的研究较为有限，仅限于 IPSEC 或 TLS 的配置管理[IETF2001b]。

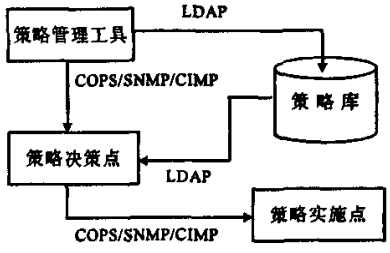


图 1－2 IETF 策略体系结构

Fig. 1-2 IETF Policy Architecture

1．3 策略的演化进程及策略系统自身管理的重要性

每个组织或企业都有着自己的主要业务目标及相应的管理策略和组织控制原则，这些主要目标可进一步分解为多个子目标，并且有着各自相应的管理策略和控制原则。这种分解和细化过程可沿着企业的组织结构继续下去。相应地，每个部门或分支机构可以根据自己的业务、功能和环境，对高层目标和策略有不同的解释，并且可以制定自己专有的目标和策略，并可以继续分解和具体化，直至某个特定的组织单位，如部门、小组等等，可以独立实现相应的子目标。在该组织

单位中，管理策略和控制原则可被进一步解释和细化成可被直接执行或实施的策略或指令。在信息和系统管理中，这些可执行的策略通常对应于对设备的功能要求和规格说明，因此可以通过管理和业务协议或约定将策略转换为物理设备的配置。这个从最初的顶层管理目标和策略的描述，一直到底层可被直接执行和实施的策略的演化过程就是典型的策略演化进程。顶层的策略是宏观的、全局的，与具体实现机制和支持设备无关的，一般可使用自然语言的形式来描述说明。而底层策略一般是具体的，与物理设备有关的，可被设备理解和执行的。由于策略的演化进程中需要对策略和需求进行不断的形式上和内容上的转换和解释，并依赖于低层现实可用的机制和功能对高层策略和需求的支持和实现，因此必须对这个过程进行一定的控制和管理[Goh1998]。

策略在信息和系统管理中的作用正越来越得到人们的重视。为达到支持企业的业务目标，从全局的角度来探讨信息和信息系统的管理，越来越多的研究和管理人员开始采用策略和高层目标来描述 IT 系统的理想行为和状态，通过对这些策略和目标的实现来完成信息和系统的管理。目前该领域的绝大多数学者将研究重点放在策略的描述语言、翻译、发布和实施机制，而忽略了一个重要的方面，即策略自身的管理以及如何满足所有相关人员对系统管理和策略的要求。

策略及策略的管理在企业的信息和系统管理中的重要性可以用一条“价值支持链”来形象地说明，如图 1－3 所示。在这条“价值支持链”中，企业的信息和信息系统支持着企业的业务，而信息和系统的管理则保证了对企业信息资源的高效利用，管理策略则指导着系统的管理，最后，策略的管理则支持对策略自身的管理，策略的管理是这条“价值支持链”的最后一环，因为策略的管理可以施加到自己本身。

图 1－3 价值支持链

Fig. 1-3 Valule Support Link

策略管理的目标和任务不仅要管理和控制策略的演化进程，还要使所产生的策略能够尽量满足所有与管理策略相关的人员的各种需要，这些相关人员包括策略的制定者和审计者、策略的实施者和维护者、以及策略的终端用户等。由于他们在策略演化过程中所扮演的角色不同，因而对策略有着不同的要求和理解，如策略的制定者希望能够以他们所熟悉的业务术语、以自然语言的形式来定义策略，

而不必涉及太多、太深入的技术细节，但同时又希望了解所定义的策略被实现的情况；策略的实施者和维护者，通常是系统管理人员，则希望策略是具体和详细的，同时又有一定的灵活性，能够让他们选择所熟悉或喜欢的方式来执行策略，并方便他们向终端用户宣传和解释管理策略；终端用户则希望策略是简单明了、易于理解的。

以上可以看出，策略的演化过程是一个系统的、完整的过程，过程的各个阶段是相互依赖、相互影响的。只重视策略的制定，而忽视策略的执行，再好的策略形同虚设；同样只重视技术而忽视策略的分析和制定，即使用有经验丰富的管理员也无法实现成功的系统管理；最后，终端用户的理解和配合也是成功的系统管理的一个重要因素。

1．4 本文的研究重点和贡献

基于策略的管理突出了策略在管理中的作用，然而在一个复杂的信息系统中，管理策略本身就构成了一个特殊的、复杂的、相对独立的子系统，并形成于一个复杂而且完整的过程，这个过程将包括策略的定义、分析、翻译、发布、执行和维护等等。因此，需要在一个系统的方法学的指导下， 将管理过程中的各个环节和各种资源有机地组织起来。在策略的形成和应用过程中，人们经常会面临一些问题，如缺乏系统的方法学的指导，无法纵观策略演化过程的全局，无法将各个管理活动有效地组织起来等等。本文将重点研究策略自身的管理，明确提出了策略工程和策略生命周期模型的概念，并给出了一个合理的、完整的策略生命周期模型，可用以指导实施基于策略的管理方法在企业的信息和系统的管理中的应用。尽管本文的研究重点是信息系统的安全管理，相信本文中的概念和方法学同样适用于其它管理领域，如网络的 QoS 管理等等。

目前基于策略的管理领域的绝大多数学者将研究重点放在策略的说明语言、翻译、发布和实施机制上。从策略生命周期模型可以看出，在策略演化过程中，最重要而又最容易被忽视的阶段是信息和系统管理需求分析，它是该领域最薄弱环节，目前尚未见有正式的研究成果。正如系统需求分析在软件开发过程中所起的作用一样，管理需求分析在成功的系统管理中起着决定性的作用。Moffett 曾建议可以将高层策略看作是一种需求，而低层策略则是对需求的一种实现[Moffett1999]，因此软件工程中的许多系统需求分析的概念和技术可以被借鉴和采用。经过对各种系统需求分析技术的研究和比较，本文采用了基于目标的需求分析技术（Goal-based Requirement Analysis），在该方法的基础上，做了相应的改进和扩展，并

提供了一个基于目标的企业信息和系统管理需求分析框架模型，使之能够很好地应用于信息管理需求分析和管理策略的确定。另外，本文还将简单介绍另外一种基于用况和误用况的管理需求分析方法，也是在研究过程中给出的另外一种较为有用的方法，可以与基于目标的分析方法结合起来使用。

策略的分析和验证也是目前该领域中的重点和难点之一。在策略演化过程中的各个阶段，都需要对相应层次的策略进行分析和验证，除包括常见的对策略的语法、语义分析，关联分析，冲突的检测和消解，还包括策略的完整性、完全性和可行性分析，以及策略的优化、合并/分解等等。由于在策略的演化过程中各个阶段所处的上下文环境不同，策略的抽象层次不同，因此，各个阶段的分析和验证的侧重点和任务也不一样，所采用的分析方法和建模工具也不尽相同。本文将分别采用彩色 Petri 网和空间几何技术，在不同的策略抽象层次和阶段，对系统和策略进行建模分析。当然，这两种技术并非是全部的、最好的技术，更多、更有效的方法,包括形式化和非形式化的验证分析技术，仍有待于去研究和探讨。

策略的翻译、发布和实施同样是基于策略的管理中的关键环节，也是目前大多数研究人员的研究重点，本文将对它们做一些简单的探讨。

1．5 论文组织方式

本论文总共由七个章节组成。

第二章概述了背景知识和相关的研究工作。本章纵览了信息安全的目标和机制，简单介绍了在信息系统开发、应用和管理中应遵循的通用的安全准则，介绍了策略的说明方法和 Ponder 策略说明语言，介绍了目前正越来越得到重视的安全工程和安全需求分析的概念，重点介绍了基于目标的系统需求分析方法和 KAOS，一种利用一阶实时线性时态逻辑来表达目标和需求的技术。

第三章叙述了本文的核心思想：信息和系统管理策略工程和策略生命周期模型的概念。策略生命周期的研究目的在于研究管理策略自身的管理，可用于指导整个基于策略的管理过程。该章给出了一个合理的、完整的策略生命周期模型，明确定义了策略演化过程中各阶段的目标、任务、活动、输入/输出和参与者，以及各阶段之间的依赖关系等等。

第四章提供了一个基于目标的企业信息和系统管理需求分析框架模型，给出了建立目标模型的步骤和方法，以及根据目标模型定义信息和系统管理策略的规则。企业信息和系统的管理需求分析是编写和定义管理策略的依据和基础，在成功的系统管理中起着决定性的作用，该章还重点讨论了在系统安全管理中，如何

理中，如何根据恶意目标建立反目标模型，并给出了根据反目标模型确定安全需求和定义安全策略等的方法。另外，该章本节还简单讨论了如何采用用况和误用用况来进行信息和系统的管理需求分析以及指导管理策略的制定。

第五章介绍了策略演化过程中的另外一个重要任务，即策略的分析和验证，该章分别在策略的高层、中间层和低层三个不同抽象层中各选出一到两项重要的、目前该领域中尚未成熟、亟待解决的一些策略分析和验证内容进行讨论。由于在策略的演化过程中各个阶段所处的上下文环境不同，策略的抽象层次不同，因此，各个阶段的分析和验证的侧重点和任务也不一样，所采用的分析方法和建模工具也不尽相同，本章将给出一些相应的新颖、有效和实用的分析方法。

第六章简单介绍了目前策略的翻译、发布和实施中常用的一些基本方法，它们是目前大多数研究人员的研究重点。

第七章是对本文所做的工作的一个总结：首先回顾和讨论本文所取得的成果，然后给出了未来的研究方向。

第二章 背景和相关研究

基于策略的管理最初应用于网络的 QoS 管理，许多标准化组织、研究机构和公司已经做了大量的研究，制定了一些相关的标准和开发出了一些初级的实用产品，如用于策略决策点 PDP 和策略实施点 PEP 之间交换策略信息的协议 COPS(Common Open Policy Language) [Durham2000]是 IETF 从 1996 年开始开发的，最初用于协商资源预留策略 RSVP[Herzog2000] 。Microsoft 和 Cisco 于 1998 年提出了目录使能的网络 DEN(Directory Enabled Network)[DEN1998]。DEN 的基本思想是把所有与系统管理有关的信息保存在目录中，管理系统使用这些信息完成管理任务。可以看出，DEN 是基于策略的管理的一个雏形。后来该研究工作被移交给 DMTF。为了使管理信息标准化，DMTF 在原有的信息模型 CIM(Common Information Model) [DMTF1999]的基础上，定义了 PCIM(Policy Core Information Model) [Moore2001]，CIM 和 PCIM 是一种面向对象的信息模型，可以被映射到特定的格式以保存在目录中，这些目录可使用（L）DAP 协议或其它协议作为访问协议。

基于策略的管理在系统安全管理的应用目前刚刚兴起，安全管理与 QoS 管理即有相同又有不同的方面，而且远远复杂于 QoS 管理。系统的安全是一个相对而不是绝对的概念，并且是与上下文相关的，系统安全的程度在某些方面是可以根据需要来设定和提供的，因此某些学者提出了 QoSS(Quality of Security Service)的概念[Henning1999, Irvine2000]，即把安全看作是一种服务质量，可用于指导安全机制的选择（如加密方法）或安全强度的选择（如密钥长度）等等。在实际应用中，系统的安全要复杂得多，QoSS 的概念只适用于其中的很小一部分。本文的研究重点是信息和系统的安全管理，本章将首先简单介绍系统的安全和安全策略的概念和相关的研究，然后逐步介绍其它背景知识和相关的研究。

2．1 信息安全和安全机制

信息安全的目的是保护信息资源，使之免遭不良企图和不可控事件所带来的不良后果。学者们一致认为信息安全是一个总体和系统工程，这是指安全并非仅仅是计算机系统的一个属性，而是包括了计算机系统及其所处的整体环境的一个属性。另外，信息安全是一个相对的，而非绝对的概念，一个安全的信息系统指的

是信息资源所面临的某种风险降低到一个可以接受的程度，而不是完全消除了该风险。

2．1．1 信息安全的目标

随着信息技术的迅猛发展和广泛应用，信息安全的目标和要求也在不断地变化和扩展。变化和扩展的主要原因是系统之间日益强大的互联特性，以及建立在强大的互联能力基础之上的新的分布式系统及其所支持的社会协作的能力。高度的分布带来了系统质的变化，因为它所支持的组织协作关系，改变了很多传统的概念和假设，如系统内部和外部之间的界限和关系等等。同样，这些高层的、与社会学和业务相关的变化也反过来要求重新评估低层支持系统的安全目标及保护机制。传统的信息安全目标，如机密性、完整性和可用性等，也因此赋予了新的内涵。下面将简单介绍信息安全的四个主要目标，另外要注意数据和信息在概念上的区别和关系：数据是信息的一种实际表达形式，信息寄宿于数据中，数据有多种形式，如数据库中的记录、计算机中的文件、IC 卡中的文件、以及纸质的文档等等。

- 机密性(Confidentiality)

机密性是信息安全的主要目标之一，要求信息资源只能由被授权的用户访问。这里的访问可以有很多种解释，如浏览、执行、占有、甚至是知道其存在与否等等。在移动计算中，机密性还包括保护可移动数据免遭抄袭、伪造、复制以及其它形式的欺诈等等。

- 完整性(Intergrity)

信息完整性的要求远比安全性复杂，信息完整性要求确保资源（包括数据和程序）免遭未授权或不适当的修改，并且系统中的数据能正确地反映它们所代表的真实世界[Samarati2000]。完整性的主要目的是防止欺诈和出错。

Sandhu 根据完整性要求的严格程度将完整性的定义依次划分为五个层次[Sandhu1993]：

1. 对数据质量的期望，数据要能满足用户对数据质量的要求；
2. 保证数据不受未授权的修改(Unauthorized Modification)；
3. 保证数据免受不恰当的修改(Improper Modification)；
4. 保障信息流的正确方向，如在 Biba 模型中，要求信息只能从高完整性对象向低完整性对象流动；

5. 数据不会被篡改，如数据在存储介质中或通讯中不会被篡改和丢失，或至少能够检测到数据的篡改或丢失。

- 可用性

可用性要求能够保证授权用户对资源的合法访问。除了要防止拒绝服务攻击（DoS）外，还要求协调共享服务的用户之间的合作。因此，可用性除涉及系统性能，还涉及系统用户的行为，以及监测错误和系统恢复等等。

- 可审计性(Accountablility)

可审计性是指系统能够保留和保护审计信息，使得影响安全的任何行为都能够被记录下来，并且能够追溯到责任方。可审计性有两个主要目标：一是审计资源的使用记录；二是对用户的审计。后者将包括组织或企业的组织结构，职员信息及职责，职员之间的协作关系等等。

在不同的应用环境中，在以上四个基本的信息安全目标基础上还可以延伸出其它的安全目标，如保护隐私(Privacy)，不可抵赖(Non-Reputation)，可匿名性(Anonymity)等等。

不同领域的企业和组织对计算机和信息安全的要求是不相同的，企业和组织应根据已有的支持业务运作的信息处理原则、目标和要求来确定相应的信息安全的要求。例如，在军事和政府部门，安全的主要目标是防止机密信息的泄漏，而在商业领域，安全的要求有所不同，尽管机密性依然很重要，但在商业环境下更注重的是信息的完整性，即防止对信息不恰当的或未授权的修改[C&W1987]。

2．1．2 信息安全机制

安全目标和安全策略定义了系统元素的理想状态和要求，需要低层具体的支持机制来实现。这些实现机制就是系统所采用的安全机制和安全服务，包括身份识别和验证、访问控制、密码学技术、审计和可信恢复等等。其中访问控制是信息安全领域中最基本也是最复杂的一个服务。许多学者认为，所有安全策略的核心是对访问信息资源的约束和控制[Crook2001]，有的学者甚至认为可以用访问控制策略（包括正态和负态，即允许和禁止两种形式）来表达所有与应用和网络相关的安全策略[Burns2001]。访问控制主要用来限制已通过身份验证的用户的活动，以控制对系统和资源对访问，并保证只有授权的、合法的访问才能发生。访问控制机制可以保护数据和信息免遭非法的泄漏和非法或不当的修改，同时向合法的用户保证数据的可用性[Samarati2000]。

传统的访问控制模型分为任意访问控制 DAC(Discretionary Access Control)和强制访问控制 MAC(Mandatory Access Control)两种，而 RBAC（Role-Based Access Control）目前在复杂系统的访问控制中得到越来越多的重视和应用。

- 任意访问控制 DAC

根据用户（可以是人或进程等等）的身份或用户所属的组来控制对目标资源的访问，之所以称为任意访问控制是由于用户可以任意把所拥有的资源访问权利传递给其他的用户。访问矩阵(Access Matrix Model)是最常见的 DAC 模型[Graham1972, Harrison1976]。在访问矩阵模型中，系统的状态由一个三元组(S,O,A)表示，其中 S 是主体 Subject 的集合，O 是目标对象 Object 的集合，A 是访问矩阵，矩阵的行对应主体，列对应目标对象，而表项 A[S,O]则定义了 S 对 O 的访问权利。在具体实现时，访问矩阵可采用权限表(Authorization Table)，访问控制列表(Access Control List)和能力表(Capability)等形式。

现在许多学者对 DAC 的模型进行了扩展。如 Brtino 等人将时态约束的概念(temporal Constraint)引入授权策略[Bertino1998]，时态约束限制了访问权限的有效时间。Samarati 认为可对授权规则附加更多的约束，如根据系统的状态、目标的状态、现有的授权状态（基于历史的授权 History-based Authorization）等对授权进行进一步的限制[Samarati2000]。大多数学者认为把授权分为正(Positive)和负(Negative)两种模态(Mode)是很有必要的[Samarati2000]。

任意访问控制存在一些缺陷，例如 DAC 不能区分用户和在系统中代表用户的主体（Subject）。访问权限被授予用户，用户登录到系统，并启动进程，该进程在系统中代表用户。DAC 系统根据用户的权限来确定进程的访问请求是否合法。一些恶意代码如特洛伊木马(Trojan Horse)便可以利用这个漏洞，对系统实施攻击。

任意访问控制不能控制信息流，一旦信息被进程获得，便失去对信息的控制能力。

- 强制访问控制 MAC

强制访问控制 MAC 主要控制信息在目标之间的流动，即信息流。最典型的强制访问控制策略是多层安全策略 MLS(Multilevel Security)，一般用于军事系统，是 TCSEC(Trusted Computer System Evaluation Criteria)的一个核心要求[DoD1985]。

在多层安全策略中，每一个主体 Subject 和对象 Object 都分配有一个访问等级类(Access Class)，访问等级类的集合构成一个偏序集(partially ordered set)，等级

之间的偏序关系定义为支配(dominance)关系，用“≥“表示。一般一个访问等级类有两部分组成：安全级别(security level)和一个类别集合(a set of categories)。安全级别构成一个全序集，如 TS(Top Secret), S(Secret), C(Confidential), 和 U(Unclassified)，其中 TS>S>C>U；类别集合是一个无序集，表示部门、领域、职能等，如 Nuclear, Army 等。访问等级类的支配关系可定义为：访问等级类 C1 支配（“≥“）访问等级类 C2，当且仅当 C1 的安全级别大于等于 C2 的安全级别，并且 C1 的类别集合包含 C2 的类别集合。

Bell-LaPadula 模型是一个经典的多层安全 MLS 模型[Bell1973a&b]，该模型定义了两条准则：

1. 简单安全特性(simple property)。一个 Subject 只能读安全等级类低于或等于它的 Object 的内容，即不能向上读(No-Read-Up)；
2. *一特性（star-property）。一个 Subject 只能向安全等级类高于或等于它的 Object 写入信息，即不能向下写(No-Write-Down)。

Bell-LaPadula 模型主要用于保证信息机密性。另一个经典的多层安全策略是 Biba 模型[Biba1977]，用于保证信息的完整性。Biba 模型中也定义了两条准则：

1. 不能向下读(No-Read-Down)。一个 Subject 只能读安全等级类高于或等于它的 Object 的内容；
2. 不能向上写(No-Write-Up)。一个 Subject 只能向安全等级类低于或等于它的 Object 写入信息。

Bell-LaPadula 模型控制信息从低向高流动，以保证机密性。Biba 模型控制信息从高向低流动，以保证完整性。如果系统既要求机密性，又要求完整性，则系统中的 Subject 和 Object 需同时拥有两个访问等级类，一个用于控制机密性，一个用于控制完整性。

强制访问控制 MAC 策略的缺点是对访问的控制过于严格，缺乏灵活性，往往不能满足实际应用的需要。

- 基于角色的访问控制 RBAC

RBAC 近年来得到了越来越多的关注，特别是在商业领域，被认为是除传统的 MAC、DAC 之外的一种选择[Samarati2000]。RBAC 能够较自然地反映企业的组织结构，能够定义和实施面向企业业务需要的安全策略。从访问控制的目的来说，确

定一个用户在组织中的责任(Responsibility)要比确认用户是谁更重要。在 DAC 中，访问控制取决于用户对信息的拥有权（Ownership），在 MAC 中，访问控制取决于 Subject 和 Object 的安全等级类，二者都不能满足确定用户责任的需要，只有 RBAC 能够根据用户在一个组织中的责任，来管理用户对资源的访问。

尽管目前存在许多 RBAC 模型[Ferraiolo1992, Taegen1995, Sandhu1996]，但基本思想是一致的，其核心概念是角色 Role。一个角色对应于一个组织中的职位或一个任务，资源的访问权限被赋予角色，而不是直接赋予用户，如果用户被允许担当某个角色，则用户便可以拥有该角色所拥有的相关权限。一个用户可以扮演多个角色，一个角色也可以拥有多个成员，如果一个用户的职位或责任发生变化，需要改变相应的权限，则只需改变其所扮演的角色即可，因而大大简化了访问控制的管理。

RBAC 除了能够简化权限管理外，还具有许多其它优点，例如，RBAC 可以直接支持三个著名的安全准则：最小权利(Least Privilege)，职责分离(Separation of Duty)和数据抽象（Data Abstraction）。对最小权限的支持是通过对角色的适当配置，使角色只拥有完成其任务所必需的权限，用户虽然被授权扮演一个角色，但只是在需要时才能激活该角色；对职责分离的支持通过在完成一项敏感任务时，规定必须由两个互不兼容的角色一起完成，既二人原则(Two Person Rule)；对数据抽象的支持，是把访问许可定义在目标对象所允许的操作或方法上，而不是直接的 read 、write、 execute 等操作。RBAC 具有较大的灵活性，完全可以仿真传统的访问控制模型，如 MAC 和 DAC[Sandhu1998, Osborn2000]。

2．1．3 权限管理和授权机制

当用户要求访问资源时，访问控制系统根据预先定义的权限集来决定是否允许用户的访问要求。权限是对资源访问的授权许可，是组织的安全和管理策略的一种具体体现。授权则是根据管理策略授予用户相应的访问资源权限的过程。尽管访问控制和授权经常被混为一体，本文认为强调它们之间的区别是十分必要的：授权涉及的是用户权限的定义，而访问控制则根据授权所定义的权限来控制对资源的访问。在一个复杂的企业信息系统中，权限的管理远比想象的要复杂得多。许多学者已注意到这个问题，并给出了一些较好的建议和方法。

在 RBAC 中，为鉴别和定义一个组织中的角色、权限以及角色和权限的关系，Epstein 和 Sandhu 提出了角色工程(Role Engineering)的概念，并给出了一个角色—权限分配模型[Epstein2001]。在该模型中，角色和权限之间加入了三个中间

层：工作(Job)、工作模式(Work Pattern)和任务(Task)。组织中的每个角色可以从事一到多种类型的工作，为完成工作，角色要实施一些活动(perform some activities)，而这些活动根据其内容和目的聚类成一个或多个集合，称为工作模式，每个工作模式包含一系列的步骤，每个步骤对应于一个任务，并由单个的用户来完成。如果在执行某个任务时，需要访问信息资源，则相应的资源访问权限就赋予该任务，并且沿着中间层向上一直传递给角色。该模型可通过自上而下和自下而上两种方法来实现。在自上而下方法种，先定义角色，再层层分解直至权限；在自下而上方法中，先鉴别资源访问权限，再层层合并，最后再定义角色。在实践中，两种方法可以同时使用，或以一种为主，另一种作为检验手段。

另外一种类似的角色—权限分配模型是 Napolean[Thomen1998]，Napolean 将 RBAC 框架分为七层，分别是对象(Object)、对象句柄(Object Handle)、应用约束(Application Constraints)、应用键(Application Key)、企业键(Enterprise Key)、键链(Key Chain)和企业约束(Enterprise Constraints)。对象是 Napolean 中最基本的组件，对应于面向对象的分析或编程方法中的类(Class)，对象只能通过所定义的对象的公共方法（Public Method）来访问，为方便访问权限的管理，这些公共方法可以组合成几个集合，称为对象句柄。应用约束定义了运行时访问对象所需要满足的条件，应用键是应用层上的角色，实质上是根据应用程序的所提供的功能和用途而定义的对象句柄的集合，一般由应用开发人员来定义。企业键是应用程序的一个具体安装中的实例化的应用键，本地系统管理员可以将企业键赋予用户，从而实现了权限分配。企业键可以进一步组合成键链，以体现组织的安全策略和约束。最后，定义在键链上的企业约束则体现了更复杂的组织的安全约束和策略。

以上两种方法都具有一定的局限性，只适用于特定的应用环境，如 Napolean 适用于基于 CORBA 和 DCOM 的应用系统，Epstein 的方法适用于有着静态结构和明确分工的组织环境中，没有考虑组织的动态特征和业务流程。

授权机制也是访问控制和权限管理中的一个研究重点，好的授权机制能提高系统的安全性，实现某些通用的安全准则，如“最小权限”、“良构的事务”等等，后面将专门介绍这些通用的安全准则。Thomas 的基于任务的授权控制 TBAC (Task-based Authorization Control) 正是这方面的一次颇有见地的尝试[Thomas1994&1997]。

Thomas 认为一个组织可以被看作是由各种活动（或任务）组成的一个复杂的网络，这些任务往往是跨部门或跨组织的。用户根据他们的角色、责任和义务被

安排和授权完成这些任务。可以把组织看作是一个系统，要求维持一定的满足完整性要求的状态。组织内部的控制(Control)和程序(Program)确保任务的实施能够使系统的完整性不会被破坏。基于任务的授权控制就是对任务的授权进行建模、分析和管理，以防止未授权的活动对信息进行非法的修改，从而破坏系统的完整性。用户对资源的访问权限只是在用户执行任务时，才被动态地授予用户。Thomas 定义了任务之间的依赖关系，如时态、语义、原子关系以及委托和撤销委托关系、职责分离关系等。TBAC 扩展了传统的基于 subject-object 的访问控制，引入了与任务有关的环境信息，使访问控制具有动态特征，访问授权与任务同步，具有及时性（Just-In-Time）的特征。

Atluri 和 Huang 提出了一种工作流授权模型[Atluri1996&1998&2000]，可用于工作流管理系统 WFMS(Workflow Management System)，实现工作流和授权流的同步。为实现工作流和授权流的同步，资源访问权限在每个任务开始时才授予用户，并且在任务结束后立即取消。该模型引入了时序约束，每个工作流中的任务都定义了一个时间段$[t_b, t_e]$，任务只有在这个时间段中才能执行，而权限只是在任务的实际执行时间段$[t_s, t_f]$和有效时间段$[t_b, t_e]$的交集中才被授予实际的任务执行者。Atluri 和 Huang 还提供了一种基于彩色时态 Petri 网的方法对工作流的授权进行建模和分析。

2.2 基本的安全准则

本节所介绍的基本的安全准则不仅适用于计算机信息系统，还同样适用于传统的基于纸质文档的办公系统。事实上很多安全准则都是从后者发展而来的。

为获得数据的完整性，防止欺诈(Fraud)和出错(Error)，Clark 和 Wilson 提出了两条准则，这就是著名的 Clark & Wilson 模型[C&W1987]：

1. 良构的事务(Well-formed Transaction)：用户不能随意操作数据，而必须以一种受到约束的、能够保证数据完整性的方式来进行；

2. 职责分离(Separation of Duties)：将关键的操作分成几个部分，并且要求由不同的人来执行，以防止合法的用户对数据进行不当的修改，避免欺诈和出错。

Sandhu 进一步总结出了九点完整性准则[Sandhu1990]，除良构的事务和职责分离外，还包括：

- 用户鉴定(Authenticated User)：用户只有在通过身份鉴定，并且被认定是合适的人选时，才能修改数据。
- 最小权力(Least Privilege)：安全领域中最早被认同的重要的准则之一，一个用户只能拥有执行所分配的任务所需要的权限，除此之外，不能拥有其它额外的权限。最小权力原则可具体分为：为保证机密性的“了解应该了解的(need-to-know)”原则，为保证完整性的“做所需要做的(need-to-do)”原则，以及“最小诱惑(least temptation)”和“及时性(just-in-time)”等原则。
- 事件重构(Reconstruction of Events)：该原则主要要求通过监测和审计不当的行为，从而起到警告、预防和恢复等作用。
- 授权委托(Delegation of Authority)：主要涉及如何在组织中获取、分发和传递权力，允许管理者和用户有一定的灵活性，可以将权力或任务委托给他所信任的人。
- 真实性检查(Reality Check)：即常说的数据一致性，数据应与所映射的外部事物是一致的。
- 操作的持续性(Continuity of Operation)：该原则要求在潜在的灾难性事件发生后，系统仍能够维持一定程度的可操作性。
- 易于安全使用(Ease of Safe Use)：系统的安全措施应该是易于执行的，复杂的、难以操作的措施反而降低了安全性。

NIST 在[NIST2001]给出了 33 条在信息系统的设计、开发和运行时应考虑的系统层上的安全准则，包括“假设外部系统都是不安全的”、“实现分层的安全”、“简单明了”、“将关键的资源和公共访问系统分离开来”、“定义和描述安全需求时应采用公用语言”、“不要采用额外的不需要的安全机制”、“确保系统开发人员接受过如何开发安全软件方面的培训”等等。

以上所列举的准则都属于较高抽象层次的安全目标或策略，在日常的系统管理中，还有很多与系统和设备有关的准则，如口令管理和登录管理原则等等，一般较为明确，可直接实施。

2．3 安全管理和安全策略

为了支持计算机和信息安全，许多安全技术被广泛地采用，如 I & A (身份识别和验证)，访问控制，审计，密码学技术，可信恢复等等。安全技术需要由合适的管理和程序来支持，否则安全技术发挥不了其应有的作用，或者当应用环境发生变化时，不作适当的技术调整，其安全作用会大打折扣，甚至完全丧失。许多安全事故都是由于管理不善引起的，可以说信息安全来自“三分技术，七分管理”，因此必须重视信息系统的安全管理[科飞2002]。

2．3．1 系统安全管理

在 OSI 的管理框架中，系统安全管理属于系统管理的五个领域之一。安全管理的任务被限定于对安全技术的管理和支持，主要包括两个方面的任务：（1）管理和保证网络和系统的安全环境，包括检测违反安全的事件和维护安全审计，（2）保证网络管理任务的实施过程的安全性[OSI1994]。

Sloman 等人定义安全管理为支持对授权策略的说明以及将这些策略转换成能够被安全机制实施的信息，以完成访问控制、密钥分发管理、安全监视和日志记录等任务[Sloman1993]。

Hyland 等人定义安全管理为对提供安全服务的应用程序进行实时监视和控制，保证在当前环境和安全策略下，为管理风险所采取的安全措施能发挥应有的作用[Hyland1998]。

IETF 定义安全管理为制定及实施关于身份验证、资源访问、以及对资源使用的审计等方面的策略[Moore2001]。

以上对安全管理和安全策略的定义偏重于对安全功能和管理和安全策略的说明和实施，属于前文对安全策略分层（图 1－1）中的第三、四层的范畴，而忽略了对高层即企业层和业务层目标对支持。信息技术和安全技术的最终目标是为企业的业务目标服务，企业的业务目标对信息安全技术的直接和间接的影响应该受到足够的重视，而这也是目前大多数研究人员的共识。

2．3．2 策略的说明

管理目标（即高层策略）描述了被管理系统和管理活动应该完成什么任务和功能，以及应该能阻止或避免什么事件的发生。这些管理目标通常以自然语言的形式给出，在进一步被处理和细化以前，它们需要被表示成一种较为正式和精确的说明形式。策略的说明有三种主要方法：

- 基于逻辑的方法

传统的基于逻辑的形式化语言可以用来表达安全策略，它们的数学背景有利于策略的分析和验证，但却难以直接映射到具体的实现机制，并且难以为人们使用和理解，如 ASL(Authorization Specification Language)[Jajodia1997]、RDL(Role Definition Language)[Hayton1998]，RSL99(Role specification Language)等等[Ahn1999]。

- 基于规则的方法

在该方法中，管理策略被说明成一系列的规则，这些规则以 "**if** Condition（条件） **then** Action（动作）" 的形式出现，可以很容易的被映射到某种实现机制，缺点是表达能力有限。

- 使用说明性策略语言(Declarative Policy Specification Language)

说明性的策略语言属于高层抽象的策略语言，如果从管理人员的角度出发，策略说明的最好的方法是使用策略说明语言，相对于其它两种方法来说，它具有相当的灵活性和较强的表达能力，如 SPL(Security Policy Language)[Ribeiro2001]、Tower[Hitchens2001]、TPL(Trust Policy Language)[Herzberg2000]、PDL(Policy Description Language)[Lobo1999]，甚至 XML 也可以用来描述策略，如 XAXML[OASIS2001]。下面将专门介绍一种较为成熟的策略语言 Ponder，本文将使用一种类 Ponder 语言来定义策略。

2.3.2.1 Ponder

英国伦敦大学皇家学院的 Moffett 和 Sloman 等人研究分布式系统的管理和基于角色的管理已有十余年了，Ponder 是他们的最新研究成果。Ponder 是一个面向对象的策略语言，用来定义分布式系统的安全和管理策略[Damianou2001&2002, Dulay2001]，Ponder 的表达能力很强，既能表达只能由人才能理解和实施的高层策略和目标，也能表达由自动代理（Automated Agent）解释执行的底层策略。

Ponder 根据策略的执行地点将策略分成两类：在目标方执行的策略和在主体方执行的策略。前者包括授权策略(Authorization Policy)、委托策略(Delegation Policy)等，后者包括职责策略(Obligation Policy)等。

授权策略定义一个主体能够对目标实施什么操作。实质上这是一种访问控制策略，用以保护资源和服务免受非授权访问。授权策略分为正(positive)和负(negative)两种模态，正态授权策略定义允许的主体对目标的操作，负态授权策略定义禁止的主体对目标的操作。一个授权策略的例子如下：

```
inst auth+ switchPolicyOps {
  subject /NetworkAdmin,
  target  /Nregion/Switches,
  action  load(),remove(),enable(),disable(),
}
```

该策略说明一个/NetworkAdmin 的成员可以对/Nregion/Switches 域中对 Switch 进行 load(),remove(),enable(),disable() 等操作。

委托策略定义用户可以将他所拥有的资源访问权限委托给别的用户，一个委托策略如下：

```
inst deleg delegSwitchOps {
  subject /NetworkAdmin,
  grantee /DomainAdmin,
  target  /Nregion/Switches/typeA;
  action  enable(),disable(),
  valid   time duration(24),
}
```

该策略说明一个/NetworkAdmin 的成员可以将对/Nregion/Switches typeA 域中的 Switch 进行 enable(),disable() 等操作委托给/DomainAdmin。valid 子句定义委托的有限期为 24 个小时。

职责策略定义了当一个事件发生时，管理员或管理代理所必须实施的行为或动作。一个职责策略的例子如下：

```
inst oblig+ loginFailure {
  on   3*loginFail,
  subject s=/Nregion/SecAdmin,
  target  t=/Nregion/Users^{userid};
  do    t disable()->s log(userid),
}
```

该强制策略说明了当用户 3 次登录失败后，/Nregion/SecAdmin 的成员，即安全管理员应该将用户帐号挂起，并将该事件记入日志中。

职责策略的实施一般由策略的主体，即管理员或管理代理完成。

Ponder 语言还给出了元策略的概念（Meta-Policy）。元策略是限制和约束策略的策略，它可以定义策略之间必须满足的约束关系。Ponder 使用 OCL(Object Constraint Language)来说明约束。例如元策略可定义职责分离（Separation of Duty）约束如下：

```
Inst meta budgetDutyConflict raises conflictInBudget(z) {
  Pa subject->intersection(Pb subject)->notEmpty    and
  Pa action->exist(act | act.name="submit")         and
  Pb action->exist(act | act name="approve")        and
  Pb target->intersection(Pa target)->OCLIsKindOf(budget),
  z->notEmpty,
```

}

该策略限制了一个职员不能既是一个预算的提交者，又是该预算的批准者。

此外，Ponder 语言还给出了其它一些形式的策略，以满足定义策略的不同需要，如信息过滤(Information Filtering)策略，限制策略(Refrain Policy)，以及 Group、Role、Relationship 等概念。

Ponder 提供了一个统一的策略定义语言，可广泛用于网络、存储、系统、应用和服务等方面管理策略等制定。同时，Ponder 的编译器也已经开发出来，并可免费获得。目前 Ponder 编译器的功能仅限于将策略编译成策略对象(Policy Object)，并进一步被翻译成防火墙规则、WINDOWS 的安全模板和 JAVA 安全策略等。

Sloman 等人的研究重点一直放在策略的说明语言和策略的实施模型上，这虽然是策略生命周期中两个重要的环节，但还远远不够。另外，策略的实施模型也较简单，一般假设授权策略由目标机器上的访问控制机制来实施，而职责策略由管理员或管理代理实施。实际上策略的实施远非这么简单，应充分考虑实施的环境，如网络的拓扑和配置信息等等。例如，授权策略“Alice 可以从家里的 PC 机登录公司的 FTP 服务器”，该策略不仅要求在 FTP 服务器上为 Alice 设置访问帐号，还要在沿途的防火墙中设置相应的过滤规则，甚至在不同的防火墙处，其过滤规则也不相同，如为了限制分组的访问路径，Alice 的访问分组在入口 A 处的防火墙的被允许通过，而在入口 B 处的防火墙则可能被设置为禁止通过。

2．4 安全工程和安全需求分析

在软件工程领域，为开发出安全可靠的信息系统，许多学者提出了安全工程的概念[Brose2001, Devanbu2000]，并建议将安全工程中的各项活动整合到软件的开发过程中，安全措施不再是系统开发的事后补救措施，而是和系统的功能一起成为系统开发的最初阶段就必须考虑的因素和设计的依据。在 SSE-CMM(System Security Engineering Capability Maturity Model)中，安全工程被定义为“由一系列的活动组成，这些活动的目的是为了了解安全风险、根据已知的风险确定安全需求、将安全需求转化为安全功能、确定安全机制正确性和有效性的可信度、确定系统中残存的安全漏洞是否是可接受的、并将各个工程领域中的有关活动有机地整合在一起”[SSECMM1999]。

安全工程的方法和概念将安全因素整合到系统开发的整个生命周期过程中，从需求分析和初步设计开始一直到软件的安装维护，由于强调了与安全相关的活动的重要性，所开发出的系统要比系统开发结束后插入的安全组件或补丁，以及其

它安全补救措施要可靠得多，同时成本也降低很多，通常只是后者的十分之一不到[Lee2002]。安全工程包含了许多与系统安全有关的活动，如 SSE-CMM 中定义了 11 个过程，包括评估安全风险、评估威胁、识别和评估系统薄弱点、确定可信参数、提供安全输入、定义安全需求、检查和验证安全等等[SSECMM1999]。这些活动主要是在高抽象层次上的、面向管理的和基于风险分析的活动。其他的学者的侧重点略有不同，如 Tettero 等人认为安全工程的重点应放在系统设计等早期的系统开发阶段的安全需求分析上[Tettero1997]。

由于系统安全问题的复杂性以及与系统开发、维护的关联性等等，决定了要系统地处理好安全问题，必须打破传统的学术研究领域的界限，将多个相关领域的科研成果有机地综合起来。这些领域将包括计算机安全、需求工程、组织行为学、组织管理、软件工程等等[Crook2002]。另外安全需求和安全策略的实现，是嵌入到系统的功能里，还是由管理系统来完成，这是一个分工和成本－效益分析权衡问题，一般说来，安全机制和安全功能可以作为系统功能的一部分，而这些安全机制的管理、配置和策略则是由管理系统来完成，这也正是基于策略的管理的核心思想所在。而且，基于策略的管理系统是一个相对独立的系统，可以运用于已有的系统之上。

Baskerville 将安全系统的设计和开发方法划分为三代[Baskerville1993]：

1. 基于检查列表的方法(Checklist-Based)。该方法允许设计者从一个可用的安全控制列表中选择合适的安全控制方法，使用风险分析来量化安全控制机制并限制设计者选择实用有效的控制。该方法是一种自下而上的方法，主要目标是为系统中的各元素选择合适的安全控制机制。

2. 基于工程的方法(Engineering-Based)。使用已有的过程模型和支持工具来进行风险分析和安全机制的选择，这也是目前所普遍采用的方法。

3. 使用抽象模型和推理的方法(Abstract Model and Reasoning)。这也是目前越来越得到人们的重视，同时也需要进一步完善的方法。系统的抽象模型和推理使得安全设计能够和用户的需求分析结合在一起，并独立于具体的实现。

安全需求分析作为安全工程中最重要的一个环节，已逐步得到研究人员的重视。Anderson 将安全需求分析定义为“定义安全策略并且征求系统拥有者同意的一个过程” [Anderson2001]。尽管可以采用传统的需求分析方法和技术，安全需求分析需要新的技术和方法的支持，例如传统的需求分析方法不能对支持安全策略的

组织程序(Organizational Program)建立有效的模型[Crook2002]。下面简单介绍几种已有的、常用的安全需求分析方法。

2．4．1 风险管理 (Risk Management)

根据 BS7799 的定义，风险管理是一个识别、评估、控制、消除或降低安全风险的活动，通过风险评估来鉴别风险大小，通过制定信息安全策略、采取适当的控制目标和控制方式对风险进行控制，使风险被避免、转移或降低至一个可被接受的水平[BSI1999]。风险管理是目前大多数国际或国家信息安全管理标准或已有的成功的实践所采用或推荐的一个基本方法[NIST1995, BSI1999, OCTAVE2003]。

在风险管理中，风险被定义为特定的威胁(Threat)利用系统资源的薄弱点(Vulnerability)，导致资源遭受损害的可能性。其可能性取决于威胁发生的可能性和系统中可被利用的薄弱点存在的可能性。而风险评估则是对威胁、后果、和薄弱点以及三者发生的可能性的评估。风险管理过程主要包括以下几个步骤：

- 了解系统的上下文环境；
- 鉴别系统资源和潜在的对资源的威胁；
- 针对潜在的威胁和系统的薄弱点，评估由每个威胁对资源所构成的风险；
- 选择解决方案，确认风险降至可接受的范围；
- 重新实现和运行系统，并开始新的评估过程。

尽管目前存在多种不同的风险管理方法，其基本方法和原理是相同的，主要区别在于风险的量化方法、对评估人员的技能的要求、评估过程面向组织而不是系统本身的程度、以及与系统设计的结合程度等等[Chivers2004]。

2．4．2 需求工程(Requirements Engineering)

将安全或者范围更广的非功能性(non-functional)需求结合进传统的需求分析过程是目前该研究领域所面临的一个挑战。下一节将专门介绍基于目标的需求分析，本节先介绍另外两种方法。

用况（Use Case）和用况图在需求工程中早已得到广泛的应用， 用况图简单直观的特点，可以很好地概括系统的功能。但用况图也有一定的局限性，它适用于对系统功能性的需求分析活动，却不适用于对系统的非功能性的需求分析，如

安全需求分析。安全需求分析除了关心所期望的系统的状态和功能，还常常关注在系统中不可接受的状态和应该避免的事件等等，这一点和功能需求恰恰相反。误用用况图[Alexander2002&2003, McDermott1999, Sindre2000&2001]，是用况图的一种扩展，可以用来进行非功能性需求分析。

误用用况(Misuse Case)是一种特殊的用况，它可以描述系统或资源的拥有者不愿意发生的情况，而入侵者或误用/滥用者希望发生的情况。误用者，与用况图中的参与者相对应，也是一种特殊的参与者，参与实施恶意或误用行为。误用用况图和用况图的表示方法是一样的，二者可以合并在一起，误用者和误用用况图可以加上特殊的标志以示区分。图 2－1 是一个简单的用况和误用用况图[Sindre2000]。

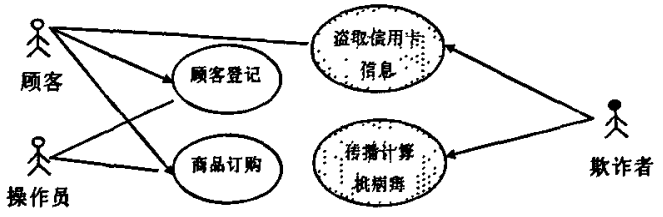


图 2－1　误用用况图

Fig. 2-1 Misuse Case Graph

图中，顾客是盗取信用卡信息的受害者，用实线相连，而欺诈者是盗用信用卡信息活动的实施者，用带箭头的实线相连。误用用况的优点是能够利用已有的常用的需求分析工具和语言，包括 UML 等；缺点是容易将用户的关注点引向对安全功能的描述，而不是简单地表达他们对安全的要求。

还有一种将安全与功能分析结合起来的方法是分别进行安全和功能分析，最后再将二者的结果合并起来，其主要思想是将安全要求定义为对系统功能的约束[Cysneiros2001, Moffett2003]。这种方法的优点是能够同时利用已有的比较成熟的功能和非功能需求分析方法；缺点是由于二者是分别进行的，在需求分析和细化过程中，无法考虑二者之间的关联和相互影响。

2．4．3 安全模式(Security Patterns)

设计模式是软件工程中支持软件重用的一种方法，设计模式描述了在某个特定的环境中会多次遇到的问题并给出了解决方案。设计模式的思想也可以被借鉴来解决安全问题[Yoder1997, Schumacher2001, SP2002]，安全模式描述了在某个特定的上下文中可能会出现的安全问题，并给出了通用的、有效的解决方案[SP2002]。一个安全模

式一般包括四个元素：名称、上下文、问题描述和解决方案或建议等[Schumacher2001]，在实际应用中，应根据模式中上下文与实际环境的相关性来选择合适的模式。安全模式的方法的优点是使用简单，可指导非安全专家去解决安全问题；缺点是过于笼统，当模式的上下文与实际环境有差别时，无法体现和适应相应的差别。

2．4．4 基于目标的需求工程(Goal-Based Requirements Engineering)

目标在系统需求分析过程的作用早已为人们所认识，基于目标的需求工程是以目标为主线，通过对目标的分析、定义、组织、精化等来获取系统的需求的一种方法，目标可以涵盖多种不同的需求，如系统的功能以及非功能需求，后者可包括安全、性能、可靠性、可维护性等等[Lamsweerde2001]。目标可以在不同的抽象层上来描述和定义，如从高层策略性的目标一直到低层的，技术性的目标等。基于目标的需求分析方法则提供了对各层次目标和需求的描述和组织的机制[Dardenne1993]。

基于目标的需求分析方法有很多优点，它提供了系统需求的完整性和相关性的标准，这一点在系统的安全管理中特别重要；目标提供了系统需求的依据；目标精化树(Goal Refinement Tree)提供了从相对稳定的高层策略性目标到低层技术性目标间的追溯能力，可以帮助分析人员向用户解释或讨论系统的需求，另外由于今天的信息系统的动态特性，要求系统的配置和管理能准确和及时地适应系统的变化，同时又能够支持高层相对稳定的目标，或者不影响其它的高层目标，这时，目标之间以及目标和需求之间的可追溯性就显得特别重要。

KAOS(Knowledge Acquisition in autOmated Specification)是 Dardenne 等人开发出的基于目标的需求分析技术[Dardenne1993, Darimont1996&1998, Lamsweerde1995]。在 KAOS 中，目标可以用一阶实时线性时态逻辑(First-order Real-time Linear Temporal Logic) 规则来表示，从而可以对目标和需求进行形式化的分析和验证。下面先简单介绍 KAOS，在第四章将会详细介绍在本文中所用到的 KAOS 中一些的概念和技术。

2．4．4．1 KAOS

KAOS 框架包括概念模型、说明语言、详化方法以及各种支持工具等[Dardenne1993]。KAOS 的概念模型如图 2－2 所示。

在 KAOS 中，对象(Object)指在应用领域中所感兴趣的事物，其实例(Instance)的状态是可变化的，对象根据性质可具体化为代理(Agent)、实体(Entity)、关系(Relationship)和事件(Event)等，对象的性质可定义为对象的属性和与其它对象的关系等。

目标是系统所要实现的目标，目标精化将目标分解为多个子目标，其中 AND－精化以“与”方式将目标分解为多个子目标，即实现目标的充分条件是同时实现所有的子目标；OR－精化以“或”的方式分解目标，即实现目标的充分条件是实现其中一个子目标即可。因此，目标精化可以以 AND/OR 的有向无环图来表示[Lamsweerde2001]。障碍是阻碍目标实现的行为或状态，是系统所不希望出现的，可以采用多种方法解决障碍，如改变目标、定义新的目标来消解或减少障碍等等。

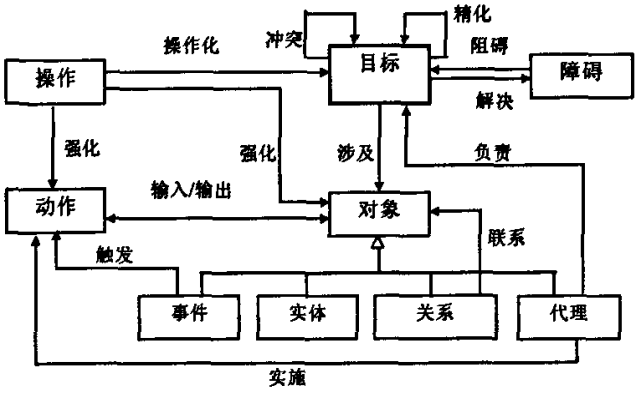


图 2－2　KAOS 概念模型

Fig. 2-2 KAOS Concept Model

目标精化的终止条件(Stop Condition)是所分解的子目标可以由某个代理单独负责实现，并称为终端目标(Terminal Goal)。终端目标即对应于系统的需求或系统假设，后者指对系统环境的假设条件，不需要由系统来实现。终端目标最后被映射到操作，这个映射过程称为目标操作化(Goal Operationalization)。一个操作以前提(pre-)、结束(post-)和触发(trigger)条件的形式定义了对象的相关状态转换。

域特性(Domain Property)是独立于系统的关于对象和操作的特性的描述，如物理定律、系统所在环境所施加的约束等等。

KAOS 语言结构分内外两层结构：外层语义层用于说明概念、属性和与其它概念间的联系；内层形式化断言(Formal Assertion)层用于相关概念的形式化定义，是可选的，可用于形式化推理和验证。KAOS 将目标分为四大类：Achieve, Cease,

Maintain 和 Avoid。Achieve 目标定义了最终要达到的状态或特性；Cease 目标定义了最终要终止的状态或特性；Maintain 目标定义了要保持的状态或特性；Avoid 目标定义的是要避免的状态。下面是一个目标的定义。

Goal Achieve [SubmittedPaperGetReviewed]
Concerns Paper, Author, Conferernce, Assistant, Reviewer
RefinedTo SubmissionPreProcessed, ReviewerSelected, PaperGetReviewed
InformalDef *所有递交的符合学术会议要求形式的论文都必须由合适的评审者进行评审。*
FormalDef ∀ au: Author, pa: Paper, conf: Conference, ∃ rvwr: Reviewer
Submitted (pa, conf) ∧ InAccordance (pa, conf) ⇒ ◊ GetReviewed(pa, rvwr)

该目标是一个 Achieve 类型的目标，涉及 Paper、Author、Conferernce、Assistant 和 Reviewer 等对象，可分解为 SubmissionPreProcessed、ReviewerSelected 和 PaperGetReviewed 三个子目标，**InformalDef** 是对目标的自然语言形式的描述，**FormalDef** 是对目标的形式化的描述，采用一阶实时线性时态逻辑规则来表示，其中“◊”操作符表示“未来某个时间”。本文将在第四章详细介绍一阶实时线性时态逻辑和所用到的操作符。

目标的鉴别和精化过程一般是一个“自上而下”和“自下而上”相结合的过程，已鉴别出的目标可以通过对“HOW”类型问题的解答而细化出其子目标；通过对“WHY”类型的问题的解答而抽象出其父目标。许多学者提供了一些经典的用于目标识别、分类、精化等的启发式问题(Heuristics) [Antón1996&1997, Lamsweerde2000]，可以用来指导建立目标模型。

为帮助建立正确和完整的目标模型，KAOS 提出了目标精化模式(Goal Refinement Pattern)的概念并提供了一个精化模式库，其中的所有精化模式都已经过逻辑验证[Darimont1996, Lamsweerde2000, Letier2001]。精化模式重用可以隐藏逻辑证明的复杂性，用户可以直接使用模式而无需对目标精化再进行逻辑上的验证。使用精化模式的另外一个用途是可以帮助分析人员发现和完善已有的不完整的目标分解。KAOS 提供的目标精化库在附录 A 中列出，例如，“里程碑模式(Mile-Stone Refinement Pattern)”是其中一个常用的精化模式，它将一个“Achieve”目标：**P** ⇒ ◊**Q** 分解成两个子目标：**P** ⇒ ◊**R** 和 **R** ⇒ ◊**Q**，其中 **R** 代表某个中间状态(◊是时态逻辑符号，表示未来某个时间点)。

操作模型是 KAOS 中另外一个重要的模型，在 KAOS 中，操作的前提和结束条件进一步分类为基本的和强化的前提和结束条件，前者定义了操作中系统基本

的状态转换，后者定义了为确保目标的实现而定义的附加条件。强化的前提、结束和触发条件的语义如下[Letier2002]：

- 强化前提条件定义了执行该操作的许可条件；
- 触发条件定义了执行该操作的职责(Obligation)，即当触发条件成立时，必须有人负责执行该操作；
- 强化结束条件定义了附加的当操作执行后必须满足的条件。

目标的操作化指确定相应操作的前提、触发和结束条件的过程，在 KAOS 中同样提供了操作化模式(Operationalization Pattern)概念和模式库[Letier2002]，参见附录 A，图 2－3 是一个“Achieve”目标 C ⇒ ○T 对应的一个操作化模式(○是时态逻辑符号，表示下一个时间点)：

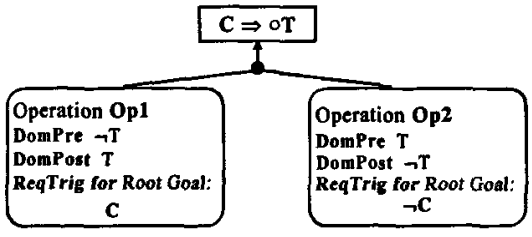


图 2－3 目标 C ⇒ ○T 对应的操作化模式

Fig. 2-3 The operationaliztion Pattern for Goal C ⇒ ○T

在目标鉴别和精化过程中产生的目标常常过于理想化，在实践中会因为意外情况的出现而难以实现，其中很多意外在系统需求分析阶段就应该进行分析和处理，在 KAOS 中，这些意外称为障碍(Obstacle)[Lamsweerde2000]。

障碍的识别可采用非形式化和形式化两种方法，非形式化方法通常使用启发式问题的方式，这些问题的形式通常是“如果存在某个事实或性质，那么就可能存在某种类型的障碍”。很多学者已经提供了一些可供参考的启发式问题，可用来指导发现障碍[Potts1995, Antón1997, Lamsweerde2000]。

除了非形式化方法，KAOS 还提供了识别和消解障碍的形式化方法。基本的障碍识别方法是一个从负目标¬G 开始的归结过程，通过选择相应的域属性理论(Domain Theory)计算实现¬G 的前提条件，每一个前提条件都可能是一个潜在的障碍。

同目标精化和操作化一样，障碍的识别也可以采用障碍识别模式，附录 A 列出了 KAOS 提供的障碍模式识别库，图 2－4 给出了“Achieve”类型目标 C ⇒

◊T 的一个典型的障碍识别模式：1－step 回溯模式。图中，◊ (C ∧□¬T)是目标 C ⇒ ◊T 的负目标，T⇒P 是一个域属性，描述了 T 的一个必要条件 P，◊ (C ∧□¬P) 就是相应的障碍。

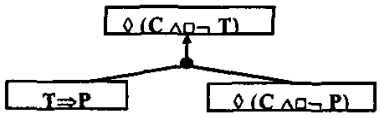


图 2－4　1－step 回溯模式

Fig. 2-4 1-step Regression Pattern

KAOS 提供了障碍的消解方法，根据所采用的策略，障碍消解方法分为三大类：消除障碍、降低障碍和容忍障碍。障碍消除方法包括改变目标、替换目标的负责代理(Responsible Agent)、防止障碍的发生等等；降低障碍指的是降低障碍发生的可能性，而不是完全消除障碍；容忍障碍包括目标恢复、降低障碍影响或者什么都不做。障碍消解将不可避免地更新目标结构，增加新目标或改变已有的目标[Lamsweerde2000]。

2．5 案例

本节给出一个具体的案例，在后面的章节中将使用该实例做案例分析，介绍和验证本文提出的概念和方法。

本案例是关于一个学术会议组委会如何组织和安排一个学术会议，使得会议得以顺利地召开，为保证会议的高水准、高质量，会议应本着公平、公正的原则选择论文评审者和录用优秀论文，防止欺诈和其它不道德的行为。除此之外，为保证会议的顺利进行，还应确保所使用的网络和信息系统的正常工作，以及相关信息的合理使用和保护。

组织学术会议的一个核心业务流程是论文的收集和评审过程，它包括 5 个主要的步骤和任务，如图 2－5 所示。

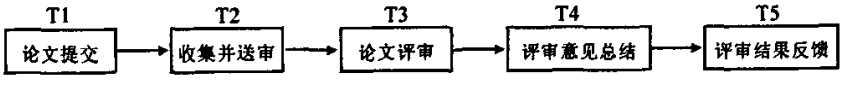


图 2－5 论文收集评审过程

Fig. 2-5 Paper Collection and Review Process

- 任务1：论文提交。作者向会议提交论文；
- 任务2：论文收集并送审。会议助理收集整理论文，并负责送审；

- 任务3：论文评审。评审者对论文进行评审；
- 任务4：评审意见总结。会议主席对论文的评审意见进行总结，并决定是否录用；
- 任务5：反馈评审结果。会议助理将评审结果反馈给作者。

作者、会议助理、评审者和会议主席都是指派给相关任务的角色，负责相应的任务的执行。

假设会议由某个科研机构的一个研究部门主持，该科研机构的内部网络如图 2－6 所示（该假设网络及所标注的 IP 地址空间与实际网络和 IP 地址没有任何直接关系，仅作为案例分析使用）。该网络包含 4 个分支，此外还有一个特殊的分支“Internet”，代表所有外部主机。这些分支由 4 个网关连接在一起。DMZ（DeMilitaried Zone）分支中有两个服务器，分别提供 http 和 ftp 服务。A (Accounting) 和 M (Management) 分支之间有一个专用连接。在 A 中有一个数据库服务器，向 A 中的所有主机和 M 中的地址为 8.0.3.11 的主机提供专用的服务——dbapp。R(Researching)为主持会议的研究部门的子网，会议的发布系统和论文在线提交系统使用 DMZ 中 Web 服务器的 http 服务向外部提供访问界面。R 中还有一个地址为 8.0.4.12 的 ftp 服务器，向所有除 DMZ 外的用户提供 ftp 服务。

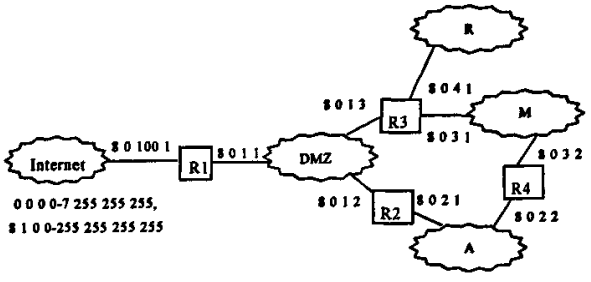


图 2－6 实例网络示意图

Fig. 2-6 A Network Example

2．6 小结

基于策略的管理方法同样可以用于信息和系统的安全管理，但由于信息安全的复杂性，使得安全策略的定义、说明、验证、实施等都远比网络的 QoS 管理要复杂得多。本章简单介绍了信息安全和管理的背景知识和相关的研究，包括信息安全目标、常用的安全机制和一些系统设计和管理中重要的、通用的安全准则。本章还介绍了策略的定义说明方法，包括基于逻辑的方法、基于规则的方法和说明

类 Ponder 语言来定义策略。由于系统安全问题的复杂性以及与系统开发、维护的关联性等等，决定了要系统地处理好安全问题，必须打破传统的学术研究领域的界限，将多个相关领域的科研成果有机地综合起来，因此许多学者提出了安全工程的概念。安全需求分析是安全工程中一个重要的任务，本章介绍了已有的常见的一些安全需求分析方法，包括风险管理、需求工程和安全模式等等。本章重点介绍了基于目标的需求分析方法，其中的 KAOS 技术采用一阶实时线性时态逻辑规则来表达目标和其它相关概念，从而可以对目标和需求进行形式化的分析和验证。本论文将以基于目标的需求分析技术和 KAOS 作为基础，通过适当的改进和扩展，使之能够很好地应用于系统安全管理需求分析。

第三章 策略工程和策略生命周期模型

策略在复杂系统管理中的重要作用已经为人们所认识，策略作为一种黏合剂将管理活动、管理工具、被管理资源以及相关人员联系在一起。基于策略的管理方法使用策略来驱动管理过程，使得管理过程能够被大大简化和自动化。基于策略的管理是一个完整的过程，覆盖了策略演化周期的所有阶段，包括企业的业务分析和系统管理需求分析、管理策略的定义、策略的分析和验证、策略的翻译、发布和实施、策略的维护等等阶段。然而在实践中，上述各阶段之间的脱节现象一直无法避免，一方面，策略的定义无法正确和完整地覆盖管理需要，而且难以被理解和执行；另一方面，在日常的管理活动中，策略可能被有意或无意地绕过或忽略。为解决这种脱节现象，必须有一个系统的方法学来指导整个基于策略的管理过程，将管理过程中的各个环节和各种策略有机地组织起来以确保管理策略的完整性(Integrity)、完全性(Completeness)、正确性(Correctness)、可实施(Applicability)和可维护性(Maintainability)。

3．1 策略工程

由于管理策略和管理活动自身的复杂性，策略的管理不再是一个简单的任务而可以看作是一项工程，为提供系统的方法学来指导和控制管理策略的整个演化过程，本文提出了一个新的概念——策略工程。

策略工程借鉴和采用工程领域的思想和观点来解决在基于策略的管理活动中所遇到的各种问题，它研究的是管理策略演化周期的整个过程，在这个过程中，策略被定义、说明、分析验证、细化、翻译、发布、执行，直至最终被废弃。整个过程可分为几个相互关联的阶段，是一个覆盖了从问题域到解域的整体性的过程。策略工程的研究范围还包括各种用来定义、执行、分析和维护策略的工具和技术，以及在策略演化过程中所涉及的各有关人员等等。

策略工程的核心是策略生命周期模型，它明确定义了策略演化过程中各阶段的目标、任务、活动、输入/输出等等，这些阶段是相互依赖而不是相互隔离的，因此策略生命周期模型还要定义各阶段之间的关系。然而值得一提的是在实际应用环境中，各阶段之间并非界限分明，而是互相渗透的，例如策略分析验证在整个

策略演化过程各阶段都有可能需要，还有策略的定义也有可能是一个反复进行的过程。

可以看出上述策略工程的定义与软件工程的定义十分相似，为了开发出高质量的软件产品，软件工程师们研究开发出了许多方法、过程、技术和工具，如有关分析和设计的方法学、项目管理、质量保证体系等等，而软件工程研究的是如何系统地组织和利用这些技术，它定义了在软件开发过程中的各项活动和相应的成果，以及如何组织这些活动，软件的生命周期也经历一系列的阶段，包括需求分析、计划、设计、实现、集成、维护和废弃等等[Schach1998]。

策略工程和软件工程之间是相互支持而不是相互矛盾的关系，一方面策略工程支持组织的信息和相关资源和活动的管理，包括软件开发过程和相关成果的管理（软件的开发过程本质上也是一个业务流程，其中间和最终产品也是一种资源）；另一方面，由于它们之间的相似性，软件工程领域的许多成熟的概念、方法、产品、技术和工具都可以应用到策略工程中，其实，策略工程中所用到的许多工具也都是软件产品。

策略工程和软件工程的一个重要的不同之处是前者通常是建立在已有的系统之上，其主要任务是有效地组织已有的管理系统和活动、高效合理地分配资源、充分利用已有的管理工具和机制、发现现有的管理策略和过程所存在的问题，定义新的策略和删除无用的策略，监督策略的执行和维护等等。

除软件工程外，策略工程还应该借鉴和采用许多其它领域的成果以应对企业的信息和系统管理的复杂性，包括组织行为学、人力资源管理、业务管理、安全工程、计算机科学和工程等等。

3．2 策略的演化过程

策略的演化过程是一个覆盖了从问题域到解域的整体性的过程，在这个过程中，策略被定义、说明、分析验证、翻译、发布、执行，直至最终被废弃，整个过程可分为几个相互关联的阶段。尽管不同的企业由于具体的需要和环境的不同，所采用的策略演化过程会有所不同，但大体上都包含以下几个阶段：

1．企业的组织结构和业务流程分析

组织结构和业务流程构成了信息系统的运行环境，是信息系统的最终服务对象，因此是在定义和实施系统管理策略时必需要考虑的因素和解决冲突时的依据。这个阶段的工作可能在软件开发过程中已经做过，企业的组织结构和业务流

程模型也已经建立。然而在策略的生命周期中所建立的模型并非简单地是这些模型的子集，因为分析人员要从中获取的是与系统安全管理有关的信息，而安全不仅仅涉及系统要提供什么功能，执行什么操作，还涉及到系统不能做什么，不能提供什么，因此必须对已有的模型和建模方法进行扩展和改进。例如，当保证对资源合法的访问的同时，还要能够阻止非法的、甚至是合法而不合适的对资源的访问。

2. 信息和系统管理需求分析

需求分析是软件开发过程中一个重要的阶段和任务，在这个阶段中，要确定出究竟需要目标产品做什么。安全管理需求分析在策略的生命周期中同样十分重要，但却有些不同，因为安全管理不仅涉及到希望系统中应该发生什么，还涉及到系统中不应该发生什么。

这个阶段的输入不仅包括第一阶段获得的企业的组织结构和业务模型，以及企业的信息和相关资源，还包括通用的安全准则和管理控制原则，如职责分离准则、最小权力准则、任务或权限委托并监督原则等等。

以上两个阶段是策略演化过程中必不可少的阶段，是正确和完整地定义信息管理策略的基础。如果管理者和系统管理员只是根据他们的经验和知识来定义策略，缺乏一个系统的方法的指导，那么所定义的策略一定是不完整和易出错的。

3. 策略的定义和说明

管理目标（即高层策略）描述了被管理系统和管理活动应该完成什么任务和功能，以及应该能阻止或避免什么事件的发生。这些管理目标通常以自然语言的形式给出，在进一步被处理和细化之前，它们需要被说明成一种较为正式和精确的说明形式。

策略的说明有三种主要方法：形式化逻辑语言、基于规则的说明语言和专用的策略说明语言。基于逻辑的策略说明方法有利于对策略的分析和验证，但却难以被直接映射到具体的实现机制，并且难以为人们所理解和使用。在基于规则的方法中，管理策略被说明成一系列的规则，可以很容易的被映射到某种实现机制，缺点是表达能力有限。最后，如果从管理人员的角度出发，策略说明的最好的方法是使用策略说明语言，相对于其它两种方法来说，它具有相当的灵活性和较强的表达能力。

4. 策略的分析和翻译

策略的分析验证主要包括：

• 语法和语义分析

策略定义必须保证在语法上的有效性，语法验证的方法取决于所采用的策略说明方法。此外，还必须检查策略的各个属性值的有效性，以及属性之间的关联的有效性。

• 策略冲突的检测和消解

不同的管理人员之间，甚至同一管理人员所定义的策略之间存在的冲突是不可避免的。冲突主要是由疏忽、错误或策略制订者之间相互矛盾的要求等引起的。另外有的策略冲突是由于定义策略的方法导致的，是允许存在的，甚至是可利用的。

• 策略的完整性验证

策略的完整性验证的一个主要任务是在高层策略被定义和说明后检查所定义的策略是否覆盖了信息和系统管理的所有要求，并且通过这些策略的执行，系统的理想状态能否被建立和维持。同样，在高层策略被翻译为低层策略后，也应该进行完整性分析，以检查它们是否覆盖了高层策略的要求。

策略翻译是一个逐步求精的过程，在这个过程中需要提供一些额外的、与所处环境有关的管理信息，如目前所采用的和可利用的管理工具、平台和程序、所涉及的系统的能力和功能、组织的结构以及人力资源信息等等。这个求精过程可以完全手工完成，也可以自动完成，但一般是由专家在计算机辅助下完成的。

在求精过程中，不同的阶段一般由不同的人员负责。同时在每个阶段，新的需求和策略经常被引入进来。

5. 策略的发布和执行

策略的发布是将精化后的策略传达到最终执行它们的实体。策略发布的一个重要任务是如何选择系统中合适的策略执行者。一个策略可由多个执行者共同或单独来执行，一个执行者一般可执行多个策略。策略发布机制必须确保所有的策略被及时和准确地传达到相关的执行者。很多现有的机制和方法可被用来发布策略，如基于 LDAP、HTTP、或 COPS（Common Open Policy Service）的方法等等。策略执行者在收到相关的策略后，开始进行相应的动作，如配置操作系统的访问控制列表、将有关任务添加到管理员的任务列表里、配置服务和设备等等，以便将被管理对象置于策略所要求的理想状态。

6. 策略监测和维护

策略监测有两个主要任务：一是检查策略是否确实得以执行，目标是否达到；二是监测环境的变化，并将变化及时通知管理者和管理员。环境的变化要求相关策略进行调整以适应新的环境。可能的环境变化包括添加了新的设备和服务、网络拓扑的变化、加入了新的人员等等，环境的变化将导致相关策略的重定义、重翻译、重新发布和执行等。

策略监测系统还负责将管理者和管理员感兴趣的一些事件记录下来，这些信息可以帮助他们评估所定义的策略和策略执行情况，以利于进行相应的改进。

策略维护也是一个重要的任务。在策略的整个演化过程中，策略被定义、执行、直至被废弃或修改，策略维护机制将记录所有这些活动的历史。这其中主要有两个目的：策略审计和策略恢复。策略是随着环境的变化而一直在更新着的，有时一个企业的信息资产在遭到非法的使用或破坏时，策略历史可被用来作为法律上的证据。策略恢复是指在采用新的策略时，由于某些原因导致系统混乱，这时需要将系统恢复到原有的配置，以便管理者和管理员去发现和解决问题。

7. 逆向工程

信息技术对一个企业来说不再仅仅是其业务的一个辅助工具，而是融入到其业务流程中，成为业务流程的一部分或一个环节，因而信息系统的安全管理不仅仅涉及到计算机信息系统的管理，还涉及到计算机系统运行所处的环境的安全。事实上，一些安全目标既可以通过强化计算机系统的安全来实现，也可以通过设计合适的组织程序和业务流程来实现，或者是二者相结合。不同的实现方式之间的取舍，需要进行评估和权衡。一个设计合理的组织结构能够大大降低信息管理的复杂度，同时一个设计合理的业务流程也往往是成功的系统管理的基础，许多著名的安全准则的贯彻，如职责分离、最小权力、良构的事务等等，都建立在合理的业务流程的基础上。因此，随着越来越多的信息技术的采用，安全需求和安全技术可能会导致一个企业的组织重构和业务流程的重新设计。

逆向工程还包括其它一些方面的内容，如在策略的分析和翻译阶段，不合适的策略的发现会导致策略的重新定义；在策略的监测和维护阶段，如果低层策略支持机制不能满足管理目标的要求，则需要引入新的机制，因此需要重新翻译策略以适应变化；同样在这个阶段，策略管理系统应该能够解释哪些策略与实现某个目标有关、或者某个管理过程或设备配置所支持的管理目标是什么等等都可以看作是逆向工程的内容。

3．3 策略的生命周期模型

从上一节的描述可以看出，策略的演化过程与软件产品的演化过程非常相似。根据软件工程的要求，软件的开发过程必须包括需求分析、说明、设计和编码、直至编译和集成。软件的设计又可以进一步分为结构设计和详细设计，前者是对产品的模块的设计；后者是对各模块内部的详细说明。在详细设计中需要选择相应的算法和数据结构。

通过比较策略和软件产品的演化过程，可以发现二者之间的一种对应关系。高层策略与软件的开发说明相对应，它们都是对要做什么的描述；中间层策略与软件产品的设计相对应，它们都是从高层目标中细化(elaborated)而来，确定了如何去实现这些目标；低层策略与软件代码相对应，它们都是对目标的具体实现。尽管这种对应关系并非十分精确，却启发研究人员在研究策略生命模型时去借鉴已有的软件生命周期模型，因为后者目前已经相当成熟了。事实上，可以把策略系统看作是一种特殊的软件产品，只是与许多大型的、复杂的软件产品相比，策略系统还是相对简单的。

目前有几种软件生命周期模型，分别适用于不同的环境和系统要求，如建造并修复模型(build-and-fix)、瀑布模型(waterfall)、快速原型法(rapid prototyping)、递增模型(incremental)、螺旋模型(spiral)、和喷泉模型(fountain)等等。由于系统管理是应用于已有的系统，因此建造并修复模型和快速原型法是不合适的，它可能会影响现有系统的正常运行，传统的系统管理方法应属于这两种类型；在系统管理中，高层策略经常被精化和翻译成多个低层策略，并且由不同的低层机制和设备来执行，而低层机制和设备常常可以独立或与其它机制合作完成多种管理目标，因此策略系统的这种特性决定了递增模型是不合适的；其它模型如螺旋模型和喷泉模型一般用于开发大型复杂的系统，而管理策略系统尽管越来越复杂，但与大型复杂的软件产品相比，还是比较简单的。

通过对各种软件生命周期模型的研究对比，本文决定选择借鉴瀑布模型来构建策略生命周期模型。基于策略的系统管理的实质是使用策略来驱动系统的管理过程，在这个过程中，策略从以自然语言形式描述的管理目标开始，一直到以设备和服务等的配置形式的低层可执行策略，这个策略演化过程非常类似与在瀑布模型中文档的演化过程，瀑布模型实质上可以看作是一个文档驱动的软件开发过程，在这个过程中，说明文档、设计文档、代码文档和其它文档如用户和操作手

册等一系列文档决定了整个开发过程。各文档之间的依赖关系与各层策略的依赖关系也十分相似。

从软件工程的观点来看，瀑布模型主要的缺陷是存在着潜在的软件产品与用户实际需要之间的偏差的风险[Schach1998]，这个风险在策略应用环境中被大大降低，这是因为策略的开发和应用主要在同一个企业内部进行，由内部管理者、管理员和员工的参与和密切配合。另外基于策略的管理的一个优点是它的灵活性，可以根据环境变化的需要而改变策略，从而改变系统的配置和行为，无需重新实现系统，这样即使系统实现和实际需要之间的偏差确实发生的话，就可以大大降低由于偏差所造成的影响。

策略的生命周期模型定义了策略从定义到实施过程中， 各个阶段所涉及到的策略的演化、管理活动、管理资源、技术和手段，以及管理活动的组织、输入、输出和所涉及的人员等等。这里给出了一个完整的策略生命周期模型如图 3－1 所示。图中过程流代表了策略生命周期中的主要步骤和活动，数据流代表的是各项活动的输入和输出，逆向过程流标识出在策略生命周期中可能的逆向工程。

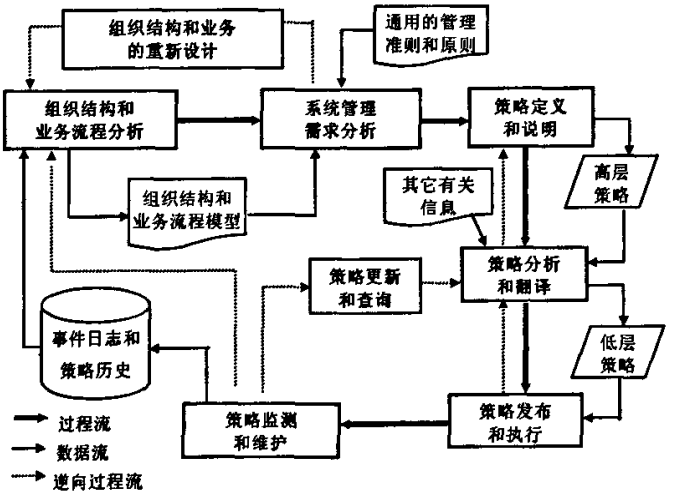


图 3－1 策略生命周期模型

Fig. 3-1 Policy LifeCycle Model

图 3－1 的策略生命周期模型中的主要步骤在上一节中已经描述过，这里不再重复。在策略的整个演化过程中主要涉及到三种人员（因简化目的，他们没有在图 3－1 中标出）：

- 策略的制定者和审计者

企业的管理者熟悉企业的结构和业务，理解企业的业务目标和规则，因此他们应该参与企业信息管理的需求分析和高层管理策略的定义。事实上，如果没有他们的支持，管理策略也难以得以贯彻执行。管理者还必须参与组织结构和业务流程的重新设计。此外，安全管理专家和系统专家也必须加入到需求分析和管理目标定义的任务中来，因为他们熟悉安全原则、管理技术、支持系统和相关资源等等，他们同时还负责策略的说明、分析、翻译和审计。

- 管理员

管理员负责策略的实际执行。他们负责日常的管理活动、维护低层支持系统、管理相关资源、监督策略的执行和信息资源的使用、向终端用户传达和解释策略，以及向管理者和管理专家汇报策略的执行情况和有关事件等等。

- 终端用户

终端用户受管理策略的影响最大，一方面，他们对企业的信息和相关资源的使用要受到管理策略的约束；另一方面，成功的信息系统的管理也需要他们的合作和支持。他们应被告知有关的策略以及相关的理由，同时他们有义务向策略的制定者和审计者反馈有关信息。

3．4 小结

由于管理策略和管理活动自身的复杂性，策略的管理不再是一个简单的任务而可以看作是一项工程，需要系统的方法学来指导和控制管理策略的整个演化过程，因此本章提出了一个新的概念——策略工程，并讨论了它与软件工程的关系。策略的演化过程是一个覆盖了从问题域到解域的整体性的过程，在这个过程中，策略被定义、说明、分析验证、翻译、发布、执行，直至最终被废弃，整个过程可分为几个相互关联的阶段，本章简单定义了这些阶段的活动和任务。最后，本章明确给出了一个完整的策略生命周期模型，定义了策略从定义到实施过程中， 各个阶段所涉及到的策略的演化、管理活动、管理资源、技术和手段，以及管理活动的组织、输入、输出和所涉及的人员等等。通过对各种软件生命周期模型的研究对比，决定选择借鉴瀑布模型来构建策略生命周期模型，瀑布模型的主要缺陷在策略生命周期模型中也得以改进和弱化。虽然策略生命周期模型的术语曾经被有的学者提及过[Wies1994]，但从未被深入展开讨论和明确定义。本章第一次对其进行了较深入的讨论，并通过借鉴已有的软件生命周期模型，给出了一个合理可行的模型。

第四章 管理需求分析和策略的制定

从本章开始，将介绍策略工程中几个重要的任务以及本文的研究成果，包括信息和系统管理需求分析、策略制定、策略的分析和验证等等。这些任务要么还未得到学术界足够的重视，要么是目前研究的难点，缺乏系统的、成熟的研究成果。本章将介绍信息和系统管理需求分析和策略制定。

信息和系统管理的主要目标是保证对企业信息资源的高效、正确和安全的利用，以支持企业的业务目标。随着信息系统规模的日益扩大和复杂化，指导和支持信息和信息系统管理的管理策略本身也越来越复杂，并形成于一个复杂的过程。编写和制定明确、全面、精确和完整的系统管理策略对于成功的企业信息资源的管理是至关重要的。而目前随机的、面向设备和技术的管理策略的编写方法已经不能满足需要，它过于依赖系统管理员的知识和经验，忽视环境对系统的要求和影响，因而往往是易于出错、不全面和滞后的，这个问题在系统的安全管理方面尤为突出。这些问题的出现要求在制定信息资源管理策略时，需要有一个系统的方法，而这个方法的起点应该是分析和确定信息资源管理的要求。

人们已经认识到无论是在系统开发还是在系统运行管理中，安全因素考虑的越早越好，在安全事故发生以后再采取补救措施、添加系统补丁、制定安全策略和增加安全控制的方法所带来的损失和增加的成本将越来越难以接受。另外，及早考虑安全因素的一个好处是管理人员可以在高层抽象层次上制定安全需求和管理策略，从而为有效和低成本地实现它们留下充足的选择余地。

Moffett 曾建议将高层策略看作是一种系统需求，而低层策略则是需求的一种实现[Moffett1999]，因此系统需求分析里的许多概念和技术可以被借鉴和采用。但系统的管理需求和系统的功能需求既有相同之处，也有不同之处，原封不动地照搬照抄软件工程里的系统需求分析技术是不可行的，需进行适当的改进。

基于目标的需求分析方法可以涵盖系统的功能需求和非功能需求，并将目标和需求通过精化过程联系起来，是目前公认的最具潜力的一种需求分析技术[Chivers2004]。本文选择基于目标的需求分析方法作为信息和系统管理需求分析和编写管理策略的一个主要方法，并在 4.1 节进行了详细介绍。当然其它的需求分析方法通过适当的改进和扩展，也可以用来分析系统管理需求，如在 4.2 节中介绍的用况和误用用况的方法。

4．1 基于目标的管理需求分析和策略制定

目标在需求分析中扮演着一个重要的角色，它表达了被研究对象应达到的目的和应具有的性质，是确定系统需求的决策依据。基于目标的需求工程是以目标为主线，通过对目标的分析、定义、组织、精化等来获取系统需求的一种方法，这种方法的优点在第二章已经介绍过。KAOS(Knowledge Acquisition in automated Specification)是 Dardenne 等人开发出的基于目标的需求分析技术[Dardenne1993, Darimont1996&1998, Lamsweerde1995]，也是本文中的管理需求分析方法的基础。在 KAOS 中，目标可以用一阶实时线性时态逻辑(first-order real-time linear temporal logic)规则来表示，从而可以对目标和需求进行形式化的分析和验证。下面首先简单介绍在本文中所用到的一阶实时线性时态逻辑里的一些概念。

4．1．1 一阶实时线性时态逻辑(First-Order Real-Time Linear Temporal Logic)

目标定义了的系统可接受的状态和状态的变化历史，状态变化历史可用系统状态的时序序列来表示，本文将采用下列时态逻辑操作符来描述系统状态的时序逻辑：

○ (下一个状态时)	● (上一个状态时)
◇ (未来某个时候)	◆ (过去某个时候)
□ (未来一直)	■ (过去一直)
W (未来一直除非)	U (过去一直直到)
$\Diamond_{\leq d}$ (在未来某个时间段d中某个时候)	
$\Box_{\leq d}$ (在未来某个时间段d中一直)	
$\Diamond_{\leq T}$ (在未来某个期限T之前某个时候)	
$\Box_{\leq T}$ (在未来一直直到某个期限T)	

在 KAOS 所采用的一阶实时线性时态逻辑中，假设时间是离散的，因此系统的状态历史(History)可以用下面的函数来定义：

h: N →State

其中 N 是一个时间点(Time Point) 的全序集合，可用自然数集来表示时间；State 是一个系统状态的集合。

下面的表达式表示系统状态历史 h 在时间点 i 时，断言 P 为真：

$$(h, i) \models P$$

其中 $h \models P$ 表示$(h, 0) \models P$，即表示在初始时 P 为真。

因此，上述时态逻辑操作符的语义可以定义如下：

$(h, i) \models \circ P$ iff $(h, i+1) \models P$

$(h, i) \models \bullet P$ iff $(h, i-1) \models P$ 且$i>0$

$(h, i) \models \Diamond P$ iff $\exists j > i \quad (h, j) \models P$

$(h, i) \models \blacklozenge P$ iff $\exists j < i \quad (h, j) \models P$

$(h, i) \models \Box P$ iff $\forall j > i \quad (h, j) \models P$

$(h, i) \models \blacksquare P$ iff $\forall j < i \quad (h, j) \models P$

$(h, i) \models P \mathcal{U} Q$ iff $\exists j > i \quad (h, j) \models Q$ and $\forall k, i<k<j \quad (h, k) \models P$

$(h, i) \models P \mathcal{W} Q$ iff $(h, i) \models P \mathcal{U} Q$ or $(h, i) \models \Box P$

$(h, i) \models \Diamond_{\leq d} P$ iff $\exists j > i$ 且 $dist(i, j) \leq d \quad (h, j) \models P$

$(h, i) \models \Box_{\leq d} P$ iff $\forall j > i$ 且 $dist(i, j) \leq d \quad (h, j) \models P$

（其中$dist(i, j)=|j-i|$，表示j和i之间的时间段）

$(h, i) \models \Diamond_{\leq T} P$ iff $\exists j, i< j \leq T \quad (h, j) \models P$

$(h, i) \models \Box_{\leq T} P$ iff $\forall j, i< j \leq T \quad (h, j) \models P$

本文还将用到下列常用的逻辑符号：

$\wedge$ (and 与), $\vee$ (or 或), $\neg$ (not 非), $\rightarrow$ (蕴含), $\leftrightarrow$ (等价), $\Rightarrow$ (永真蕴含), $\Leftrightarrow$ (永真等价)，且

$P \Rightarrow Q$ iff $\Box (P \rightarrow Q)$　　$P \Leftrightarrow Q$ iff $\Box (P \leftrightarrow Q)$

4．1．2 目标模型和目标精化(Goal Model and Goal Refinement)

目标建模的目的在于调查用户和系统管理的目标和要求、探讨满足这些目标和要求可能的解决方案。目标模型应包含目标本身内在的特征，如目标的类型和属性，以及目标之间或与需求模型中的其它元素之间的关系等等。

目标类型　目标分类可以用来帮助确定和精化目标、获取需求以及对目标和需求进行一致性和全面性检查等等。目标的分类有很多种，如功能型目标（指系统应提供的功能和服务）和非功能性目标（指所期望的系统的质量，如安全、性

能、灵活性、互操作性、可维护性等等）；硬目标（通过验证的、必须要达到的目标）和软目标（含糊的、可供选择的目标）等等。

本文主要采用了 KAOS 中的一种主要分类方法，根据目标所描述的行为的时序特征，目标可分为“Achieve”、“Cease”、“Maintain”、“Avoid”四种类型，其各自所包含的时序行为如下：

Achieve: $P \Rightarrow \circ Q, P \Rightarrow \Diamond Q, P \Rightarrow \Diamond_{\leq d} Q, P \Rightarrow \Diamond_{\leq T} Q$

（下一个时间点、或未来某个时间点、或未来某个时间段内或时间点之前要达到的状态）

Cease: $P \Rightarrow \circ \neg Q, P \Rightarrow \Diamond \neg Q, P \Rightarrow \Diamond_{\leq d} \neg Q, P \Rightarrow \Diamond_{\leq T} \neg Q$

（下一个时间点、或未来某个时间点、或未来某个时间段内或时间点之前要终止的状态）

Maintain: $P \Rightarrow Q, P \Rightarrow \Box Q, P \Rightarrow Q \ W \ R$

（未来要保持的状态）

Avoid: $P \Rightarrow \neg Q, P \Rightarrow \Box \neg Q, P \Rightarrow \neg Q \ W \ R$

（未来要避免出现的状态）

目标属性　目标的属性通常包括目标的名称、目标的描述等等，另外一个可选的属性是目标的优先级，通常作为目标冲突的消解依据。

目标链接(Goal Link)　目标链接用来定义目标之间以及目标和需求模型中的其它元素之间的关系。目标之间的链接描述了目标之间的支持或矛盾关系。目标精化是一种支持链接，包括 AND－精化和 OR－精化；目标冲突链接是一种矛盾链接，它表示实现其中一个目标将导致另外一个目标无法实现。

目标与其它元素之间的链接包括目标与代理之间的责任链接，目标与对象之间的涉及链接，目标与障碍之间的阻碍链接等等。

原始的目标通常以自然语言的形式来描述，为了能够进一步分析和处理目标、获取需求，目标应该以精确和正式的方式来说明，同策略的说明方法相似，目标可采用非形式化的、半形式化的和形式化的方法等。本文采用了 KAOS 中的目标说明语言和方法，参见 2.4.4 中的介绍。

目标精化是将抽象的、宏观的目标分解为一系列具体的、微观的子目标的过程，其中 AND－精化链接将目标链接到一个子目标集，只有该子目标集的所有子目标都达到了，才意味着达到原目标；OR－精化也将目标链接到一个子目标集，该子目标集中任意一个子目标达到了，原目标即可满足。

- 目标 AND－精化的语义如下：

在域 Dom（这里以域的属性集 Dom 代替域）中，如果下面两个条件满足的话，则称目标集$\{G_1, G_2, ..., G_n\}$是对目标 G 的 AND-精化：

1. $\{G_1, G_2, ..., G_n, Dom\} \models G$
2. $\{G_1, G_2, ..., G_n, Dom\} \not\models false$

条件 1 是 AND－精化的的充分条件，条件 2 指$G_1, G_2, ..., G_n$的合取与 Dom 是兼容的，即它们可以同时为真。另外 AND－精化有一个最小精化集的特性，即

3. $\forall i \in [1..n], \{\wedge_{j \neq i} G_j, Dom\} \not\models G$

● 目标 OR－精化的语义如下：

在域 Dom（这里以域的属性集 Dom 代替域）中，如果下面两个条件满足的话，则称目标集$\{G_1, G_2, ..., G_n\}$是对目标 G 的 OR-精化：

1. $\forall i \in [1..n], \{G_i, Dom\} \models G$
2. $\forall i \in [1..n], \{G_i, Dom\} \not\models false$

条件 1 是 OR－精化的的充分条件，条件 2 要求G_i与 Dom 是兼容的。另外 OR－精化有一个附加特性，即完全性(Completeness)，其要求如下：

3. $\forall j \neq i, \{G_j, Dom\} \not\models G \quad i \in [1..n]$

4．1．3 障碍模型和障碍精化(Obstacle Model and Obstacle Refinement)

从语义上讲，目标定义了对象或环境的理想行为，而行为则以一个状态的时序序列来描述。相反，障碍则定义了对象或环境的非理想的行为，并阻碍了目标的实现，因此障碍的反面(Negation)则定义了实现目标的必要条件。障碍的定义如下：

在域 Dom（这里以域的属性集 Dom 代表域）中，一个障碍 O 阻碍了目标 G，当且仅当

1. $\{O, Dom\} \models \neg G$
2. $\{O, Dom\} \not\models false$

条件 1 指目标的负面（Negation of the Goal）$\neg G$是障碍 O 和域 Dom 的逻辑结果；条件 2 指障碍 O 是与域 Dom 兼容的，即该障碍在域 Dom 中是可发生的。

障碍的一个附加特性——域完全性(Domain Completeness)，其定义如下：

在域 Dom 中，目标 G 的一个障碍集$\{O_1, O_2, ..., On\}$对于 G 来说是完全的，当且仅当

$\{\neg O_1, \neg O_2, ..., \neg On, Dom\} \models G$

该条件指如果障碍集$\{O_1, O_2, ..., On\}$中任何一个障碍都不存在的话，目标即可实现。障碍的这个特性在障碍识别中可以帮助我们确认是否全部障碍都已经识别出来，这一点在安全领域尤为重要。

和目标一样，障碍也可以被精化，其中 AND－精化以“与”方式将障碍分解为多个子障碍，即满足障碍的充分条件是同时满足所有的子障碍；OR－精化以“或”的方式分解障碍，即满足障碍的充分条件是满足其中一个子障碍即可。同样，障碍精化可以以 AND/OR 的有向无环图来表示[Lamsweerde2001]。障碍精化的语义与目标精化的语义相似。

●障碍 AND－精化的语义如下：

在域 Dom（这里以域的属性集 Dom 代替域）中，如果下面两个条件满足的话，则称障碍集$\{O_1, O_2, ..., O_n\}$是对障碍 O 的 AND-精化：

1. $\{O_1, O_2, ..., O_n, Dom\} \models O$
2. $\{O_1, O_2, ..., O_n, Dom\} \not\models false$

条件 1 是 AND－精化的的充分条件，条件 2 指$O_1, O_2, ..., O_n$的合取与 Dom 是兼容的。另外障碍 AND－精化也有一个最小精化集的特性，即

3. $\forall i \in [1..n], \{\wedge_{j \neq i} O_j, Dom\} \not\models O$

● 障碍 OR－精化的语义如下：

在域 Dom（这里以域的属性集 Dom 代替域）中，如果下面两个条件满足的话，则称目标集$\{O_1, O_2, ..., O_n\}$是对障碍 O 的 OR-精化：

1. $\forall i \in [1..n], \{O_i, Dom\} \models O$
2. $\forall i \in [1..n], \{O_i, Dom\} \not\models false$

条件 1 是 OR－精化的的充分条件，条件 2 要求O_i与 Dom 是兼容的。另外障碍 OR－精化也有一个附加特性，即完全性(Completeness)，其要求如下：

3．$\{\neg O_1, \neg O_2, \ldots, \neg On, Dom\} \models \neg O$

注意障碍 OR－精化的完全性与前面介绍的目标 G 的障碍集的完全性的区别。

4．1．4 基于目标的信息和系统管理需求分析(Goal-Based Requirements Analysis for Management of Information and Systems)

在进一步探讨信息和系统管理需求分析之前，下面先介绍它与软件工程中系统需求分析之间的主要区别。

- 目的不同

信息和系统管理需求分析目的在于调查企业的组织结构、业务目标及流程、信息资源和信息基础设施等，确定与信息资源的使用相关的组织控制原则、管理程序、责任和管理需求等，制定相应的管理策略和约束，以确保整个企业的信息资源的合理、高效和安全的使用。而软件需求分析的目的相对较为简单，其主要焦点在于待开发的系统和对系统功能和性能上的要求。

- 范围不同

信息和系统管理需求分析必须以全局的观点来考察整个企业的业务和信息资源，以及与企业有各种业务往来的其它企业、组织和客户等等。而软件需求分析仅限于待开发的系统和与之有直接交互的环境因素。

- 起点和方法不同

软件需求工程师首先要通过与客户交谈、评估原有的系统、调查系统所要支持的业务和阅读与系统功能要求有关的文档等来获取软件需求。而信息和系统管理需求分析首先需要分析组织的结构和业务、评估现有的管理措施和程序，调查企业目前和将要使用的信息基础设施等等。当然二者之间的考察内容有很多重叠的部分。

4．1．4．1 上下文和基础

在一个企业中与信息资源直接相关的组织元素如图 4－1 所示，它勾廓了信息和系统管理需求模型的上下文。

每个企业都有自己明确的业务目标，这些业务目标由预先明确定义的业务流程来完成。每个业务流程由一个或多个互相依赖的任务组成，每个任务都完成了某个业务目标中的一个或多个子目标。每个任务都由一个或多个基本的活动组成，每个任务或活动在执行过程中可能会消耗和/或产生信息，并使用由信息系统或其

它设施提供的服务和功能等。这些活动必须由所规定的合法的角色来完成，每个组织中的人员都分配了一个或多个角色，在实际运行时，对于每个具体的活动都从其合法的角色中挑选出一个合适的人选来实际完成它。

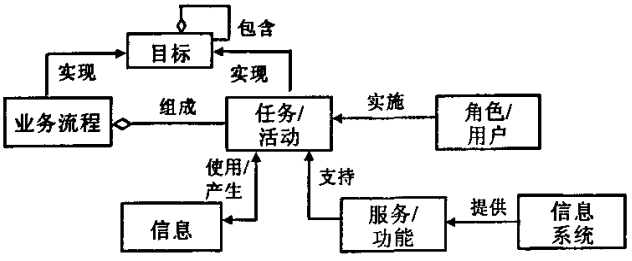


图 4－1 信息和系统管理需求模型的上下文

Fig. 4-1 The Context of the Requirements Analysis Model for Management of Enterprise Information and Systems

目标的识别通常不是一个简单的任务，图 4－2 进一步详细描述了业务目标、业务流程和任务之间的关系，指导分析人员从考察业务流程中的任务出发去获取目标。每一个任务的执行都是为了完成某个特定的目标，而一个业务流程中所有任务对应目标的实现则意味着该业务流程所支持的业务目标的实现，而业务流程和任务在一个企业中通常已经有了明确的定义，可以直接为分析人员所利用。

考察完业务目标和业务流程，还要继续考察为支持这些任务的执行，一个企业所采用或将要采用的信息系统和相关基础设施，以及它们所提供的服务和功能等等。一种服务或功能可以支持多个任务，同样一个任务可能使用不同的系统所提供的多种服务，服务或功能与任务之间的支持关系也因此需要确定下来。

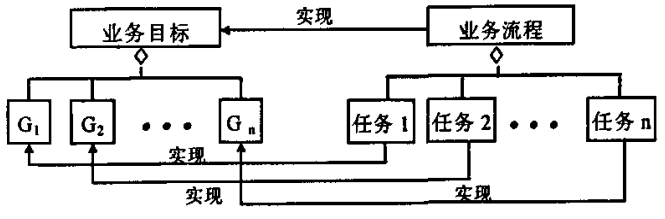


图 4－2 目标、业务流程和任务之间的关系

Fig. 4-2 The Relationship among Goal, Bussiness Process and Task

域特性(Domain Property)在目标的分析和精化过程中起着重要的作用，域特性指不随系统状态改变而改变的，总是成立的或客观存在的，系统或环境的特性[Letier2001]，如物理定理、常识和客观规律等等。域特性一般以与对象或系统相关

联的不变式的形式来表达。域特性的确定是一个逐步完善的过程，在需求分析的各个阶段都有可能需要确定有关的域特性。

4．1．4．2 基于目标的企业信息和系统管理需求分析模型

图 4－3 给出了基于目标的企业信息和系统管理需求分析模型，该模型以 KAOS 的需求分析模型为基础，进行了适当的改进和扩展。

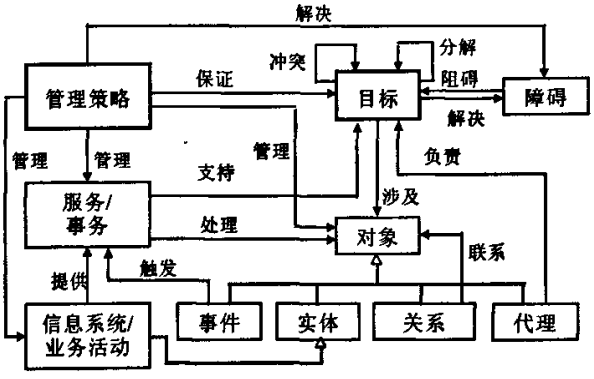


图 4－3 基于目标的企业信息和系统管理需求分析模型

Fig. 4-3 Goal-based Requirements Analysis Model for Management of Enterprise Information and Systems

在图 4－3 中，目标、障碍、对象、事件、实体、和关系的概念同 KAOS 中的基本一样，而代理却稍有不同，仅仅指活动(Active)或自治(Autonomous)的代理，如人、软件代理、进程等等，相当于 RBAC 中角色的概念。服务/事务(Transaction)组件中的服务指为实现某个特定的目标而使用的系统提供的服务和功能，系统这里泛指各种信息系统，如软件、设备、办公设施等，或者指抽象的实体，如组织单位、团体等等。事务指各业务活动中，为完成某个特定目标而整合在一起的动作的集合，它构成了具有明确的意义或目标的活动的最小单位。

管理策略是该模型中一个最重要的组件，它将模型中所有其它的组件联系在一起。管理策略管理或指导系统的配置，指导对系统提供的服务和功能的使用，保护和管理各种对象，因而确保了目标的实现。

功能/事务可能由事件触发，并且将操作（输入、输出）对象。代理在权限和能力的许可下，可以使用系统提供的服务和功能，或执行某个预先定义的事务来实现所负责的目标，其行为必须在策略的约束之下。

4．1．4．3 建立目标模型的步骤和方法

建立目标模型的步骤和 KAOS 中目标的建模和分析步骤大体相同，主要包括以下几个活动：

一、 初始目标识别

如 4.1.4.1 所述，可以从考察业务流程中的任务着手来识别初始目标。需求分析人员首先应研究与业务和任务有关的各种信息，这些信息可以从多种渠道获得，如与相关人员的交谈、文档、情景分析等等。初始目标的识别可通过对一些启发式问题的回答来进行，如“这个任务的目的是什么？”，“任务中所实施的每个动作的目的是什么？”，“为什么执行该操作？”等等。另外，如[Antón1997]中所建议的所有与动作有关的词汇(Action Words)或如[Sandia2005]中所建议的意图词汇(Intentional Words)都可能对应着潜在的目标。还可以采纳[Antón1997]中所提供的用于目标识别和分类的启发式问题(Heuristics)。在对一个任务的分析中所识别出的目标都可以看作是该任务所对应的目标的子目标。

命名每一个识别出的目标并给出简明扼要的描述，确定每个目标的类型，如果可能的话，应根据目标的类型和时态模式给出初步的目标形式化的定义。

对于图 2－5 所示的论文收集和评审过程，可以从它所包含的 5 个任务中得到 5 个初始目标：

- **T1**: 论文提交

G_1: Goal Achieve [PaperSubmitted]

InformalDef 在最后提交期限deadlineS以前，论文的作者能够通过在线论文提交系统提交论文。

FormalDef ∀ au: Author, pa: Paper, conf: Conference

Submitting (au, pa, conf) ⇒ $\Diamond_{\leq deadlineS}$ Submitted (au, pa, conf)

- **T2**: 论文收集并送审

G_2: Goal Achieve [SubmittedPaperGetReviewed]

InformalDef 所有提交的并符合会议要求的论文都能送交合适的评审者评审.

FormalDef ∀ au: Author, pa: Paper, conf: Conference, ∃ rvwr: Reviewer

Submitted (au, pa, conf) ∧ InAccordance (pa, conf) ⇒ ◊ GetReviewed (pa, rvwr)

- **T3**: 论文评审

G_3: Goal Achieve [ReviewResultReturned]

InformalDef 评审者评审论文，并且在期限deadlineR之前将评审意见返回。

GetReviewed (pa, rvwr) ⇒ $\Diamond_{\leq \text{deadlineR}}$ ReviewResultReturned (pa, rvwr)

- **T4**: 评审意见总结

G_4: Goal Achieve [ReviewResultSummarized]

InformalDef 在期限deadlineD 之前，会议主席应总结论文的评审意见，决定是否录用。

FormalDef ∀ pa Paper, ∃ rvwr Reviewer

ReviewResultReturned (pa, rvwr) ⇒ $\Diamond_{\leq \text{deadlineD}}$ ReviewResultSummarized (pa)

- **T5**: 反馈评审结果

G_5: Goal Achieve [ReviewResultForwarded]

InformalDef 会议助理将评审结果反馈给论文的作者，并确保作者及时收到评审结果。

FormalDef ∀ pa: Paper, ∃ *ass:* Assistant

ReviewResultSummarized (pa) ∧ ReviewResultForwarded (pa, ass) ⇒

◊ ReviewResultReceived (pa)

二、 目标的分析、精化和目标结构的建立

目标结构的建立是一个结合了“自上而下”和“自下而上”两种方法的过程[Lamsweerde2001]。在传统的非形式化的方法中，高层目标通过对已有目标的“WHY”类型问题的提问和回答来获得，它给出了初始目标的背景和原因，并且通过精化，可以帮助鉴别出可能在识别初始目标时所忽视的其它同层次的目标；低层目标通过对已有的目标的“HOW”类型问题的提问和回答获得。[Antón1997]中也提供了一些用于目标精化和细化(Elaboration)的启发式问题。

在这里，目标精化过程的终止条件和 KAOS 中的略有不同，必需满足下面两个条件：

- 精化后的子目标可以落实到一个或多个代理来负责，每个代理能够单独实现该子目标；
- 精化后的子目标在实现过程中只需要使用系统提供的单个服务或功能，或者只需要实施单个事务(Transaction)。

满足上述要求的目标即为终端目标(Terminal Goal)。

目标细化后所建立的目标结构可以以一个 AND/OR 的有向无环图来表示，其中叶子结点，即终端目标有两种类型，一种对应于系统的需求，另一种对应于域特性。叶子结点是定义系统管理策略的依据，下一小节将会详细介绍。

同 KAOS 一样，如果目标的形式化定义已经给出的话，形式化技术可以用来进行目标分析和细化，逻辑证明技术可以用来验证目标分解的正确性。例如目标 G_2 可 AND－精化为三个子目标，如图 4－4 所示，可以通过逻辑证明来证明该精化的正确性。

假设：　　那么：

P: Submitted　　G_2: $P \wedge R \Rightarrow \Diamond Q$

R: InAccordance　　G_{21}: $P \Rightarrow \Diamond S$

Q: GetReviewed　　G_{22} $S \wedge R \Rightarrow \Diamond T$

S: PaperCorrectlyReveived　　G_{23} $T \Rightarrow \Diamond Q$

T: ReviewerSelected

证明：

∵	$S \wedge R \Rightarrow \Diamond T$	G_{22}
	$P \Rightarrow \Diamond S$	G_{21}
∴	$P \wedge R \Rightarrow \Diamond T$	
∵	$T \Rightarrow \Diamond Q$	G_{23}
∴	$P \wedge R \Rightarrow \Diamond Q$	G_2

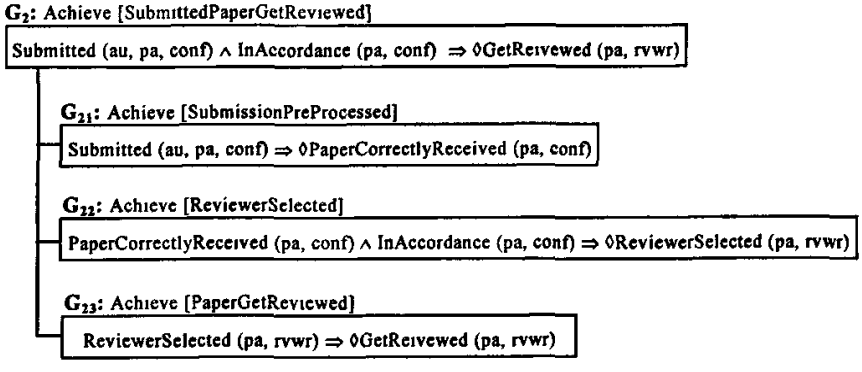


图 4－4 目标 Achieve [SubmittedPaperGetReviewed]的 AND－精化

Fig. 4-4 The AND-Refinement of Goal Achieve [SubmittedPaperGetReviewed]

同样，KAOS 中的精化模式(Refinement Pattern)也可以直接用来精化目标，其优点在第二章中已经介绍，精化模式的选择方法是一个尚待解决的问题，目前还主要依靠分析者的经验和对系统的了解，或者通过对各种方案的成本、性能和效果的比较而确定。例如，通过对目标 G_1 应用[Darimont1996]中提供的精化模式 RP5（参加附录 A），可以得到三个子目标，如图 4－5 所示。

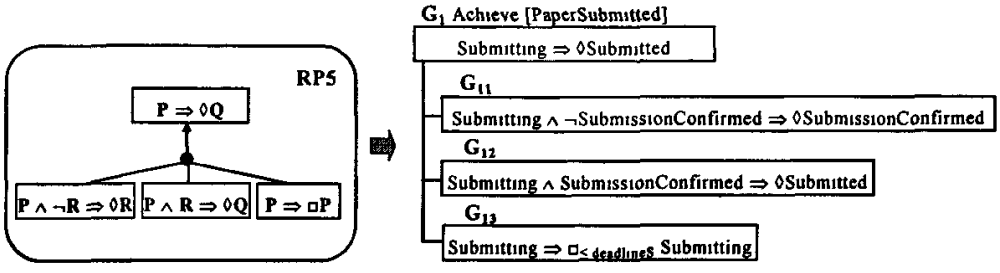


图 4－5 应用精化模式 RP5 精化目标 G_1

Fig. 4-5 Refining Goal G_1 Using Refinement Pattern RP5

G_{11} 指提交系统应该对论文的提交进行确认，G_{12} 可以看作是一个域特性，指只有作者收到提交确认后才算论文提交成功，G_{13} 要求在提交论文成功前作者没有取消论文提交。

图 4－6 所示的是 G_2 完整的目标结构，共有 7 个终端目标。根据目标精化的结束条件，当子目标能够由单个代理负责实现，并且只需要使用系统的单个功能或服务，或者通过执行单个事务来完成，则精化结束，该子目标即为终端目标。例如目标 G_{2211}：Achieve [ReviewerConstraintRequested]可安排由会议助理负责，并且可以通过发送一封询问评审者约束的 Email 来实现，因而是一个终端目标。目标 G_{223}：Achieve [ProperReivewerSelected]也是一个终端目标，因为它由会议助理负责，并且根据预先规定的选择原则来选择评审者，选择过程可以看作是一个完整的事务。

目标精化还可以采用代理驱动的方法[Letier2002]。该方法通过考察代理的能力和对实现目标的要求，如对可监视的变量（即对象的属性），和可控制的变量的要求等等，将目标逐步分解为可被代理实现的子目标。

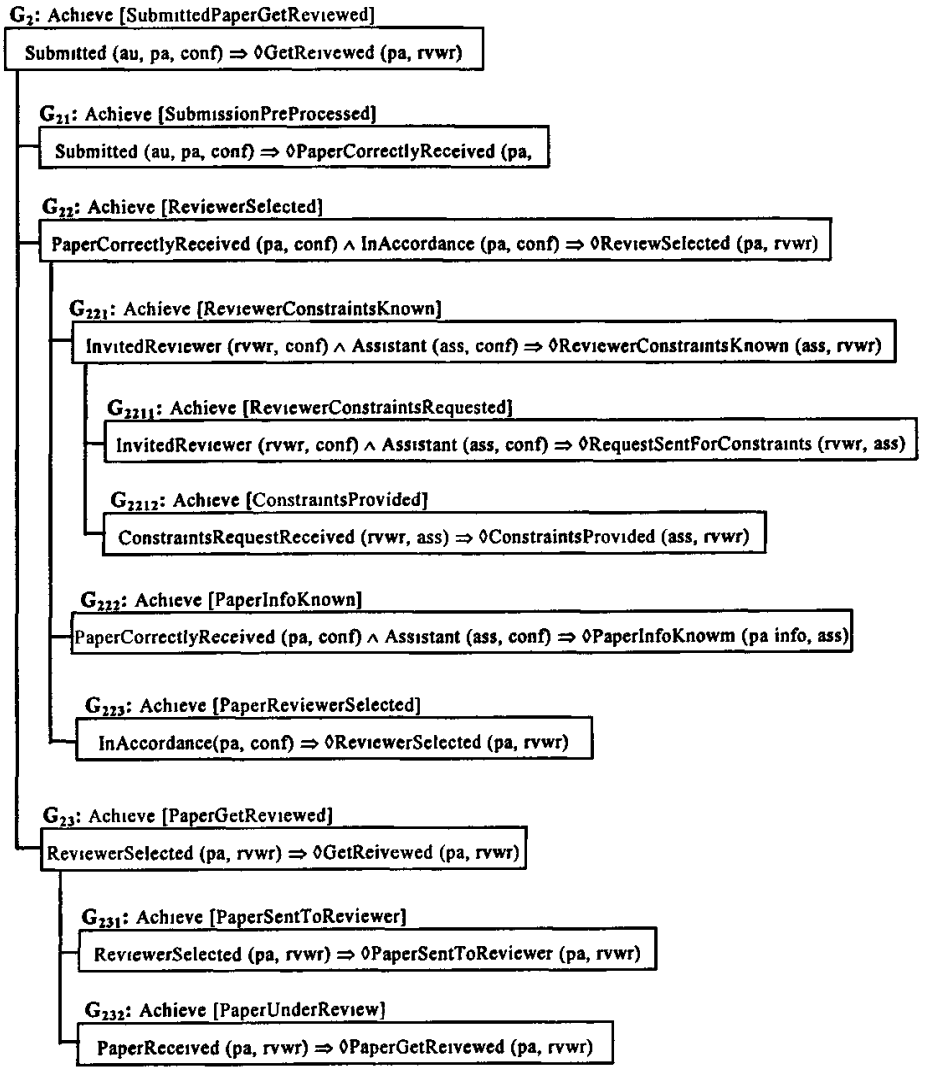


图 4－6 G_2的目标结构

Fig. 4-6 The Goal Structure of G_2

三、 确定对象和代理责任

在目标细化的同时，目标所涉及的对象也可以逐步确定下来。当终端目标确定下来后，服务/事务与对象之间的“操作”关系也可以确定下来，负责目标的代理与目标间的“负责”关系也可以初步明确下来。例如，对象 Paper（论文）可以很容易地从很多目标中鉴别出来，而 Paper 的一个属性 Paper.info 可以从目标 G_{222}: Achieve [PaperInfoKnown]中鉴别出来，Paper.info 包含了论文的作者、领

域、关键词等等信息，是会议助理选择评审者的基本依据。G_{222} 由会议助理负责，通过论文提交系统的查询服务来获得 Paper.info 的信息。

四、 确定强化前提条件和触发条件

该步骤对应于 KAOS 中的目标“操作化”，在信息和系统管理需求分析中，分析人员只对实现目标的强化前提条件和触发条件感兴趣。强化前提条件表达了为确保目标的实现而附加的各种约束和要求，如时间、地点、环境和系统的状态的约束等等。触发条件表达了在某个事件发生后，代理被要求采取相应的动作，当然前提是该代理被允许这样做，并且约束条件也能同时满足。强化前提条件和触发条件的确定可采用 KAOS 提供的相应的确定规则和操作化模式（参见附录 A），例如，对于 G_{2211}： Achieve [ReviewerConstraintsRequested]，其强化条件如下：

ReqPre for ReviewerConstraintsRequested

$InvitedReviewer(rvwr, conf) \wedge Assistant(ass, conf') \wedge conf = conf'$

上述条件要求评审者必须是该会议的受邀评审者，而会议助理必须是同一会议的助理，这是因为该科研部门可能会主持多个会议。

再如对于 G_{2212}：Achieve[ConstraintsProvided]，可得到一个触发条件如下：

ReqTrigger for ConstraintProvided

ConstraintRequestedReceived (rvwr, ass)

该触发条件描述了评审者收到由会议助理发来的 Email 询问其约束的事件。

4．1．4．4 根据目标模型制定信息和系统管理策略

管理策略将管理需求分析模型中所有其它的组件联系在一起，当目标模型建立起来后，管理策略也可以相应地确定下来。管理策略不论何种类型，通常包括四个基本部分：S、A、O 和 C。其中 S 代表主体(Subject)、A 代表相关的动作或操作(Action)、O 代表操作的对象(Object)、C 代表约束(Constraint)，说明了策略的有效条件。这四个部分均可以从目标模型中获取。

一、定义授权策略(Deriving Positive Authorization Policy)

根据目标精化的终止条件，每一个需求终端目标和部分域特性终端目标均可以安排由一个或多个代理单独负责。为实现目标，代理可能要使用信息系统提供的服务或者执行某个规定的事务，因此，代理首先应获得使用服务或执行事务的授

权。对每一个上述的终端目标都对应着一个潜在的授权策略。策略的约束也同样可以从目标说明中确定下来，目标的强化前提条件即是对策略的有效性的约束。授权策略定义规则如下：

策略定义规则 1：

responsible_for (*ag, goal*) ∧ **supports** (*sev/trans, goal*) ∧ **provides** (*sys/activity*， *sev/trans*) ∧ **operates_on** (*sev/trans, obj*) ∧ **ReqPreForGoal** (*goal, const*) ⇒ **Authorized** (*ag, sys/sev, obj, const*)

为实现目标 *goal*， 负责代理 *ag*（由负责关系 responsible_for 确定）被授权使用由 *sys* 提供的 *sev* 服务或执行某个活动 *activity* 所包含的 *trans* 事务，前提是满足 *const* 条件。操作对象由 operates_on 关系确定。函数 ReqPreForGoal(*goal, const*)通过 *const* 返回目标 *goal* 的强化前提条件，该条件是在上一小节的步骤四中确定的。

例如，根据目标 G_{2211}: Achieve [ReviewerConstraintsRequested]，可以定义一个授权策略：

```
auth+ RequestForConstraints {
    subject     a=Assistant;
    target      r=Reviewer;
    action      EmailSystem/SendEmail (ConstraintRequest);
    when        InvitedReviewer(r, conf) and Assistant (a, conf') and conf=conf';
    }
```

会议助理被授权发送 Email 给会议评审者询问其约束条件，如时间和数量限制等等。

二、定义职责策略(Deriving Obligation Policy)

目标的触发条件表达了在特定的事件发生后，代理的应采取相应的动作以完成其职责，因此，定义了触发条件的目标对应了一个潜在的职责策略，定义规则如下：

策略定义规则 2：

responsible_for (*ag, goal*) ∧ **supports** (*sev/trans, goal*) ∧ **provides**(*sys/activity, sev/trans*) ∧ **operates_on** (*sev/trans, obj*) ∧ **ReqPreForGoal** (*goal, const*) ∧ **ReqTriggerForGoal** (*goal, trigger*) ⇒ **Obligation** (*trigger, ag, sys/sev, obj, const*)

当事件 *trigger* 发生后，代理 *ag* 应使用由 *sys* 提供的 *sev* 服务或执行某个活动 *activity* 所包含的 *trans* 事务对 *obj* 进行相应的处理，前提是满足 *const* 条件。ReqTriggerForGoal (*goal, trigger*)通过变量 *trigger* 返回目标的触发条件。

例如，根据目标 G_{2212}: Achieve [ConstraintsProvided]，可以得到一个授权策略，同时还得到一个职责策略：

```
auth+ ProvideConstraints {
    subject      r=Reviewer;
    target       a=Assistant;
    action       EmailSystem/SendEmail (ConstraintOfReviewer);
    when         InvitedReviewer(r, conf) and Assistant (a, conf') and conf=conf';
    }
```

受邀评审者允许发送 Email 给会议助理，告知他的约束条件，前提是二者是同一会议的评审和助理。

```
oblig ProvideConstraintsImmediately {
    subject      r=Reviewer;
    target       a=Assistant;
    on           ConstraintRequestReceived (a, r);
    do           EmailSystem/SendEmail (ConstraintOfReviewer);
    when         InvitedReviewer(r, conf) and Assistant (a, conf') and conf=conf';
    }
```

当收到会议助理发来的询问时，评审者必须向助理提供其约束信息。

有时，根据目标得到的职责策略所要求采取的动作已经由系统实现，并由系统自动完成，这时不再需要定义该职责策略，但相应的授权策略应保留下来，因为它可能影响到系统其它组件的配置。

4．1．4．5 目标障碍分析及处理

这里将障碍的分析和处理从目标的分析中独立出来，仅仅是为了叙述的方便。事实上和 KAOS 建议的一样，它应该和目标的分析结合在一起，同时进行。KAOS 中的障碍识别和消解的方法同样也适用于本文，只是在障碍的处理方面，还可以采用其它一些针对性的方法：

- 定义禁止策略（负态授权策略 Negative Authorization Policy）以阻止障碍的发生

在某些特定的条件下，由于代理的误用或滥用信息资源而导致障碍的产生，因此可以直接定义禁止策略阻止代理的这种行为，从而消除了障碍。

- 定义职责策略规定采取相应的措施以降低障碍发生的影响

当无法避免障碍或成本太高的话，可定义相应的职责策略，要求在障碍发生后采取适当的措施以减小影响或恢复到系统可接受的状态。根据障碍定义责任策略的方法和前面根据目标定义责任策略的方法略有不同，责任策略的主体(Subject)

一般和制造障碍的代理不同，而策略要求采取的动作应该是降低障碍的影响，触发事件和条件应和障碍的触发事件和条件相同。

● 业务流程的重新设计

合理的业务流程是成功的系统管理的基础，在没有合适的消除障碍的方法情况时，应考虑重新设计业务流程或重新设计组织结构。

障碍的识别和处理的主要步骤和方法如下：

一、障碍的识别

和目标一样，障碍的识别也可以采用非形式化方法和形式化方法，在第二章中已简单介绍过，这里主要介绍障碍识别的形式化方法中的归结法。

归结法的基本思想是：对一个特定的目标 G，首先对 G 取反，得到 G 的反态¬G，在目标所在的域中，采用与 Dijkstra 的前提条件演算(Precondition Calculating) [Gries1981]相类似的方法，寻找能够实现¬G 的前提条件，每一个前提条件都对应一个潜在的障碍，具体方法如下[Lamsweerde2000]：

1. 对目标 G 取反，得到：
   O := ¬G
2. 选择合适的域规则（域特性）
   A ⇒ C
   其中 C 与 O 中的某些文字 L 相匹配，且 L 均以正态形式出现在 O 中。
3. 进行代换
   取 μ := mgu (L, C);
   O := O[L/A. μ]

其中：

- mgu (F1, F2)是 F1 和 F2 的最一般的合一；
- F μ 表示对 F 进行合一；
- F[F1/F2]表示用 F2 对 F 中的 F1 进行代换。

归结法的关键在于域规则的正确定义和选择，这取决于所分析的应用领域和分析者的经验。归结过程可反复进行直至所识别出的障碍足够精确和有明确的含义，其终止条件为[Lamsweerde2000]：

- 可以很容易地鉴别出能够使障碍为真的情景(Scenarios)；
- 能够找出合适的解决障碍的方法。

例如，对于目标 G_{223}: Achieve [ProperReviewSelected]

Inaccordance (pa conf) ⇒ ◊ ReviewerSelected (pa, rvwr)　　(G_{223})

使用归结法识别目标障碍的步骤如下：

1. 对目标 G_{223} 取反，得到 $\neg G_{223}$

Inaccordance (pa, conf) ∧ □ (¬ReviewerSelected (pa, rvwr))　　($\neg G_{223}$)

2. 选择域规则 Dom

ReviewerSelected (pa, rvwr) ⇒ProperReviewer(pa, rvwr) ∧ Convenient(rvwr, conf)　　(Dom)

该规则的一个等价式 Dom'

¬(ProperReviewer(pa, rvwr)∧Convenient(rvwr, conf))⇒¬(ReviewerSelected (pa, rvwr))　　(Dom')

该规则描述了评审者被选择评审某个论文的一个必要条件，即评审者是该论文的一个合适的评审者，且目前是方便的。

3. 进行归结代换

Inaccordance (pa, conf) ∧ □ ((¬∃ rvwr: Reviewer) ProperReviewer (pa, rvwr)

∨ ¬Convenient (rvwr, conf))　　(O)

4. 精化已初步识别出的障碍

上述的障碍可经过 OR－精化为两个子障碍：

Inaccordance (pa, conf) ∧ □ ((¬∃ rvwr: Reviewer) ProperReviewer (pa, rvwr))　　(O_1)

Inaccordance (pa, conf) ∧ □ ((¬∃ rvwr: Reviewer) Convenient (rvwr, conf))　　(O_2)

障碍 O_1： NoProperReviewerAvailable 描述了该论文没有合适的评审者的情况，这可能是由于没有邀请与该论文的领域相同的评审，或者该领域所邀请的评审与论文的作者有牵连因而不适合评审该论文。

障碍 O_2：ReviewerInconvenient 描述了合适的评审者目前不方便评审该论文，例如他可能已被要求评审几篇论文，没有精力再评审更多的论文了。

为处理上述两个障碍，可以定义一个职责策略如下：

```
oblig InviteMoreReviewer {
    on          NoReviewerAvailable;
    subject     Chair;
```

do　　InviteNewProperReviewer (pa);
}

事件 NoReviewerAvailable 指上述障碍 o_1 和 o_2.所描述的情况，当无法为论文安排合适的评审者时，会议主席应立即邀请新的合适的评审者。

障碍的域完全性(Domain Completeness)可用来验证所识别的障碍的完全性，也可以用来帮助识别未识别出的障碍，以确保所有的障碍都已被鉴别出来，这一点在安全领域特别重要，其方法如下[Lamsweerde2000]：

1. 对于域 Dom 中的目标 G 和已经识别出的一组障碍$\{O_1, O_2, ..., O_k\}$，取 $O^* = \Diamond(\neg GC \wedge \neg OC_1 \wedge \neg OC_2 \wedge ..., \wedge \neg OC_k)$，其中 OC_i 指该障碍产生的条件；
2. 检查 O^* 与 Dom 的兼容性，即二者能否同时为真；
3. 如果 O^* 与 Dom 是不兼容的，说明障碍$\{O_1, O_2, ..., O_k\}$是完全的；
4. 如果 O^* 与 Dom 是兼容的，说明障碍是$\{O_1, O_2, ..., O_k\}$是不完全的，可对 O^* 进行归结或精化处理，直至满足前面所给的终止条件，则所得到的新的障碍就是剩余的障碍。

例如，对于目标 G_{21}：Achieve [SubmissionPreProcessed]

$$Submitted (pa, conf) \Rightarrow \Diamond PaperCorrectlyReceived (pa, conf) \qquad (G_{21})$$

对目标取反，得到$\neg G_{21}$：

$$\Diamond (Submitted \wedge \Box \neg PaperCorrectlyReceived) \qquad (\neg G_{21})$$

域规则 D_1，和根据 D_1 归结所得到的障碍 O_1 如下：

$$PaperCorrectlyReceived \Rightarrow PaperComplete \qquad (D_1)$$

$$\Diamond (Submitted \wedge \Box \neg PaperComplete) \qquad (O_1)$$

域规则 D_2，和根据 D_2 归结所得到的障碍 O_2 如下：

$$PaperCorrectlyReceived \Rightarrow PaperPrintable \qquad (D_2)$$

$$\Diamond (Submitted \wedge \Box \neg PaperPrintable) \qquad (O_2)$$

检查$\{O_1, O_2\}$的完全性，其步骤如下：

$$O^* = \Diamond (\neg GC \wedge \neg OC_1 \wedge \neg OC_2) \qquad (O^*)$$

$$= \Diamond (\text{Submitted} \wedge \Box \neg \text{PaperCorrectlyReceived} \wedge \text{PaperComplete} \wedge \text{PaperPrintable})$$

事实上如果一个所接收到的论文是完整且可打印的，则算是成功地接收到了，即

$$\text{Submitted} \wedge \text{PaperComplete} \wedge \text{PaperPrintable} \Rightarrow \text{PaperCorrectlyReceived}$$

因此可以断定 O^* 是与现实不兼容的，因而$\{O_1, O_2\}$是目标 G_{21} 的一个完整的障碍集。

和 KAOS 一样，障碍精化模式可以用来进行障碍识别和精化，障碍精化的结束条件和上述归结终止条件相同。例如，对目标 G_4：Achieve [ReviewResultSummarized]应用 KAOS 中提供的饥饿模式(starvation pattern) 可以得到一个障碍如下：

$$\text{ReviewResultReturned (pa)} \wedge \Box_{\leq \text{deadlineD}} (\neg \text{ReviewResultSummarized (pa)}$$

$$\vee \exists (\text{pa}' \neq \text{pa}): \text{ReviewResultSummarized (pa')}$$

该障碍描述了因为有太多的论文及评审意见，会议主席无法在期限 deadlineD 之前全部总结完的情况，根据相关的规定会议主席可以委托会议助理帮助总结评审意见，因此得到下面的委托策略：

deleg DelegSummarizeResult {
subject Chair;
grantee Assistant;
target rvwrult=ReviewResult(pa)
operation SummarizedReviewResult (rvwrult);
when ReviewResultReturned (pa) $\wedge \Box_{\leq \text{deadlineD}}$ ($\neg$ReviewResultSummarized (pa)
$\vee \exists$ (pa' $\neq$ pa): ReviewResultSummarized (pa')
}

当无法在 deadlineD 之前完成评审意见总结时，会议主席可委托会议助理帮助完成评审意见总结。

二、障碍处理

除了前面介绍的直接根据障碍定义策略的方法外，KAOS 中的障碍处理方法也适用于本文。障碍处理将导致原有目标的改变和/或新的目标的定义，并重新定义新的相应的管理策略。这里简单介绍几种常用的处理方法，其余的方法参见[Lamsweerde2001]：

- 恢复原有目标 (Goal Restoration)

目标恢复指定义一个新的目标，要求当阻碍原有目标的障碍的条件为真时，在某个可接受的时间延迟内，原有目标得以重新满足，新的目标形式如下：

$$G^*: OC \Rightarrow \Diamond_{\leq d} G$$

其中 OC 指该障碍产生的条件。例如对于前面所讨论的障碍 NoProperReviewerAvailable，目标恢复将产生一个新的目标 G^*：

$$\text{Inaccordance (pa, conf)} \wedge \Box (\neg\exists \text{ rvwr: Reviewer}) \text{ ProperReviewer (pa, rvwr)} \Rightarrow$$
$$\Diamond_{\leq d} (\exists \text{ rvwr. Reviewer}) \text{ ProperReviewer (pa, rvwr)} \qquad (G^*)$$

该目标可 AND-精化为两个子目标：

$$\text{Inaccordance (pa, conf)} \wedge \Box ((\neg\exists \text{ rvwr: Reviewer}) \text{ ProperReviewer (pa, rvwr)}) \Rightarrow$$
$$\text{oNotifyConfChair (pa, ch: Chair)} \qquad (G^*_1)$$
$$\text{(NoProperReviewerNotified (pa, ch)} \Rightarrow \Diamond_{\leq d} ((\exists \text{ rvwr: Reviewer}) \text{ ProperReviewer (pa, rvwr)} \qquad (G^*_2)$$

第一个子目标 G^*_1 指当论文 pa 没有合适的评审者时，会议助理应立即通知会议主席，因此可以相应地定义一个职责策略，和一个授权策略授权会议助理这样做。第二个子目标 G^*_2 要求 pa 的合适的评审者在 d 期间内应该是可得的，因此可以定义一个职责策略要求当收到会议助理的通知时，会议主席应尽快邀请新的合适的评审者，当然还有一个相应的授权策略允许这些做。

- 预防障碍的发生(Obstacle Prevention)

为避免障碍的发生，可定义一个新目标要求障碍发生的条件永远不会为真，其形式如下：

$$G^*: \Box \neg OC$$

同样对于障碍 NoProperReviewerAvailable，应用障碍预防措施可得到一个不同的解决方案，产生一个新的目标防止障碍的发生：

$$G^*: \Box \neg OC$$
$$\Box \neg(Inaccordance \wedge \Box (\neg ProperReviewer))$$
$$\Box (\neg Inaccordance \vee \Diamond ProperReviewer)$$
$$\Box (Inaccordance \rightarrow \Diamond ProperReviewer)$$
$$Inaccordance \Rightarrow \Diamond ProperReviewer$$

新的目标要求事先要为会议所涉及的每个领域邀请足够多的评审者，但这可能会导致有的评审者没有论文可评审，这是因为没有该领域的论文提交，或相对于该领域的评审者的数量来说，实际提交论文的数量太少。对于同一障碍

NoProperReviewerAvailable，与障碍预防相比，目标恢复方法可能会产生一些延迟，但成本较低。具体应选择哪一个方法，取决于成本－效益分析或其它因素。

- 降低障碍的影响

当障碍发生后，为降低障碍的影响，可定义一个可接受的最低目标要求，其形式如下：

$$G^{*}: OC \Rightarrow G'$$

其中 G'指要满足的最低要求目标，或是原目标 G 的双亲目标。例如对目标 G_3 Achieve [ReviewResultReturned]取反，可直接得到一个障碍如下：

$$\text{GetReviewed (pa, rvwr)} \wedge \neg\Box_{\leq \text{deadlineR}} \text{ReviewResultReturned (pa, rvwr)} \qquad (O)$$

可定义一个弱化后的目标取代原有的目标：

$$\text{GetReviewed (pa, rvwr)} \Rightarrow \Diamond_{\leq \text{deadlineR}} (\text{ReviewResultReturned (pa, rvwr)} \vee \text{DelayNotified\&Explained(pa, rvwr)}) \qquad (G_3')$$

该目标要求评审在 deadlineR 之前将评审意见返回，或者如果不能按时返回的话，应通知延迟并解释原因。因此可以得到下面一个职责策略：

```
oblig Notify&ExplainDelay {
    subject     r=Reviewer;
    target      a=Assistant;
    on          time=deadlineR;
    do          Notify&ExplainDelay(pa)
    when        GetReviewed(pa, rvwr) and ¬ReviewResultReturned (pa, rvwr);
    }
```

- 减少障碍的发生

该方法并不是要彻底消除障碍，而是减少障碍发生的概率。该方法主要针对负责人员的工作疏忽或滥用职权而导致的障碍。例如对于目标 G_{13}：

$$\text{Submitting} \Rightarrow \Box_{\leq \text{deadlineS}} \text{Submitting} \qquad (G_{13})$$

直接对目标取反，得到一个障碍如下：

$$\Diamond (Submitting \wedge \neg\Box_{\leq \text{deadlineS}} Submitting) \qquad (\neg G_{13})$$

该障碍描述了作者在论文提交失败后忘记或不愿再重新提交的情况，为减少这种情况的发生，可定义一个职责策略：

```
oblig RemindReSubmit {
    subject     SubmissionSystem;
    target      au=Author;
    on          SubmissionFailure (au);
    do          EmailSystem/SendEmail(RemindOfReSubmission);
}
```

当有论文提交失败的情况发生时，论文提交系统应发送 Email 给作者，提醒作者重新提交。

4．1．5 信息和系统安全管理需求分析

为了强调信息和系统安全管理的重要性、特殊性和复杂性，本文使用了一个单独的章节来讨论它。

一个现在被广泛接受的观点是信息和系统的安全在系统需求分析阶段的开始就应该系统地进行。安全不仅仅是信息系统的一个属性，而应该是整个系统所在的环境的一个属性。因而，有些安全目标可能与信息和系统没有直接的关系，而是涉及了业务中的社会方法学 [Chivers2004]。这意味着并非所有的安全目标都可以从基于信息资源的分析中确定，如风险分析等。基于目标的分析方法却能很好地适应这种情况，这是因为目标分析可以覆盖所有与企业的信息和系统直接和间接相关的内容，包括业务流程、组织结构、业务规则、当然还有信息资源等等。

信息安全有三个主要目标，分别是机密性、完整性和可用性。机密性所关心的是信息的秘密和隐私方面的要求，要求信息资源只能被授权的主体访问。这里的“访问”可以有多种解释，包括浏览、执行、占有、甚至是知道资源的存在与否。完整性关心的是保证信息的精确性和完整性，阻止对信息非法的或不恰当的修改。可用性指保证授权的用户对信息的合法访问。

在一个理想的世界中，如果每个人都遵纪守法，不会做任何未授权的事，没有恶意代理，也不存在资源的滥用，那么安全问题将不再重要。取而代之的是系统的性能和无故障方面的要求，以防止非理想状态的出现。“安全(Security)所涉及的是故意的行为，……非故意行为涉及的是性能和故障问题(safety)，而不是安全” [Schneier2003]。在这个理想的世界中，机密性问题不复存在；完整性问题仅仅涉及数据的一致性和数据质量的要求；而可用性问题则仅仅涉及信息的提供和服务质量的保证以及故障处理等。这些质量、性能和故障处理问题在上一节中的目标和障碍分析中均已处理。然而在现实世界中，安全问题却是无法回避的，需要进

行特殊的对待，这是因为存在着各种各样的恶意代理，有着各种各样的恶意目标。

安全问题涉及的是故意行为的实质启发分析人员从恶意代理的角度去建立反目标模型，考察分析它们的反需求(Anti-Requirement)[Crook2002]。反目标模型(Anti-Goal Model)的目的在于对恶意代理的行为和能力进行建模，考察它们所能够执行的操作和所能监视和控制的变量，因而针对恶意代理的行为和能力所应该采取的措施则表达了系统对安全的需求[Lamsweerde2004]。

有的反目标可以很容易地直接识别出，但大多数反目标却是很难直接识别或预测的，主要是由于它们的多样性和模糊性，这时可以从基本安全目标的反目标着手，正如使用归结法识别目标障碍一样。下面将分别讨论这两种情况。

4．1．5．1 根据恶意目标建立反目标模型

一些与业务相关的恶意目标可以从与目标识别的来源相同的文档、场景分析和交谈等中直接鉴别出。事实上，一些“Avoid”目标如果从相反的角度来看的话，就是一种恶意目标。如在本文的实例分析中，对论文的评审过程有下面的要求：

论文的评审过程应该是公平的，质量差的论文不能通过不诚实的途径获得通过。

在传统的目标分析中，从上述声明中可以直接得到目标“Avoid [PoorPaperGetAccepted]”。如果从恶意代理的角度来看待这个问题的话，可以得到一个对应的反目标“Achieve[PoorPaperGetAccepted]”。从反目标着手分析而不是从目标和障碍分析来获取安全需求，往往是一种直接、有效和全面的方法，同时还可以直接利用目标分析中的方法和工具。

当初始反目标识别出来后，可以采用和目标的分析和精化一样的方法进行处理，如可以采用“HOW”和“WHY”类型问题的提问和回答对反目标进行细化，也可以通过目标精化模式来细化。恶意目标的负责代理、所要使用的系统服务和功能、所执行的事务，所涉及的对象，以及所受到的约束等等，同样都可以在反目标的细化过程中确定下来。

Achieve [PoorPaperGetAccepted] 细化后的目标结构如图 4－7 所示，注意图中的子目标 Achieve [IntrudingIntoSystem] 不是一个终端目标，可以继续精化为一个称为“威胁树 (Threat Tree)”的子树，由于该精化太过于面向技术和具体化，所以不在目前的需求分析阶段进行讨论。

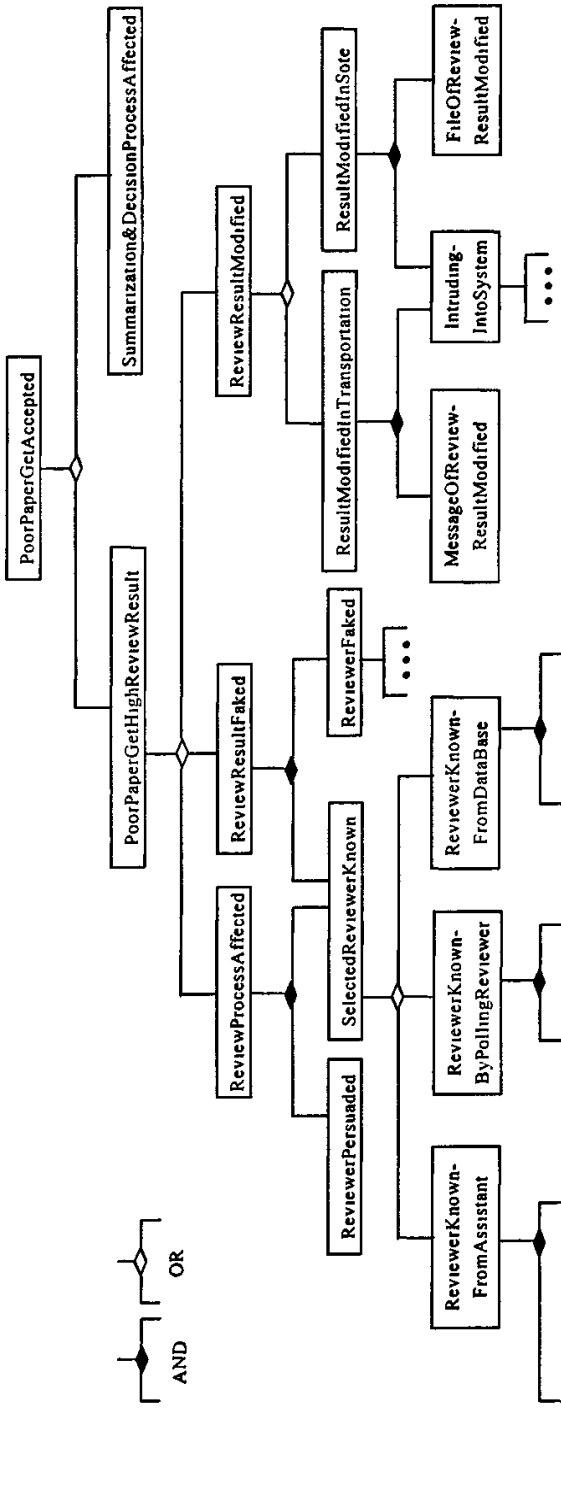


图 4-7 反目标 PoorPaperGetAccepted 的精化结构

反目标和目标模型分析的主要区别在于根据模型所定义的管理策略的方法是不同的。根据反目标定义策略的方法主要由下面两种：

1．定义禁止策略(负态授权策略 Negative Authorization)阻止恶意目标的实现

为实现（恶意）目标，（恶意）代理需要能够监视一组变量，并且能够以某种方式控制另外一组变量，这里变量对应于所涉及对象的属性。为阻止恶意代理达到其恶意目标，可以保护这些变量，使其对恶意代理是不可见或不可控的。因此可以定义相应的禁止策略使得反目标无法实现，定义规则如下：

策略定义规则 3：

responsible_for (*ag, anti-goal*) ∧ **supports** (*sev, anti-goal*) ∧ **provides**(*sys/activiy, sev/trans*) ∧ **operates_on** (*sev, obj*) ∧ **ReqPreForGoal**(*anti-goal, const*) ⇒ ¬ **Authorized** (*ag, sys/fun, obj, const*)

当约束条件 *const* 成立时，禁止代理 *ag* 使用系统 *sys* 提供的服务 *sev*，或执行事务 *trans*，对对象 *obj* 进行相应的操作。

例如对于图 4－7 中的反目标 Achieve [ReviewerPolled]，可定义一个禁止策略（负态授权策略），禁止作者向评审者询问所评审的论文：

```
Auth- PollReviewers{
    subject      Author;
    target       Reviewer;
    operation    PollForPapersReviewed (conf: Conference);
    }
```

当然，并非反目标结构中的所有目标都是恶意目标。大多数情况下，一个恶意代理可能同时是一个普通的代理，并负责某些普通的目标，他可能会滥用这些职权来达到恶意目标。这时可采用两种方法来处理这种情况，一是将恶意代理负责普通目标作为一种客观事实，视为一种域属性；二是同样定义禁止策略，以限制对权限的滥用，但禁止策略则会与原先的授权策略相冲突，这时可采用冲突消解策略来处理，这里不对其展开讨论。但应该指出的是在消解冲突时，（反）目标结构应作为决策的依据，例如，如果一个与授权策略相冲突的禁止策略是根据某个反目标定义的，而该反目标是高层反目标的一个 AND－精化子目标，则可以考虑给该禁止策略较低的优先级，或简单地删除即可。

例如对于图 4－7 中的目标 Achieve [ReviewerListKnown]，可以相应定义一个禁止策略禁止作者获知评审者的名单。然而该策略可能与原有的策略相矛盾，如

评审者名单可能在“会议征稿”中已经公布，因此，目标 Achieve [ReviewerListKnown]可以看作是一个事实（域属性），而不是一个反目标。

2．定义职责策略降低恶意目标的影响

如果根据反目标所定义的禁止策略难以实现，或者无法有效地阻止恶意行为，或者因为与其它策略相冲突而无法得到执行，此时可以采用在目标障碍处理中所使用的一些方法，如定义职责策略，规定有关代理采取合适的动作以降低恶意行为的影响和恢复理想的状态等等。

例如，对于反目标 Achieve [ReviewerPolled]，前面已经定义了一个禁止策略 **Auth- PollReviewers{...}**，很明显，该策略在现实中很难有效地得以执行，因此需要另外定义一个职责策略来降低恶意目标的影响或发生的可能性：

```
oblig ReportPolling {
    subject        Reviewer;
    target         Assistant;
    on             PollingForPapersReviewed (au);
    do             ReportPolling(au);
}
```

当作者询问所评审的论文时，评审者应向会议组织者报告此事。该策略还可起到一定的威慑作用。

3．重新设计业务流程

在上述两种方法仍无法有效阻止恶意行为，或成本太高，可以考虑重新设计业务流程，这时可将所遇到的问题考虑进去，使新的业务流程不再受该问题的困扰。

例如对于反目标 Achieve [Summarize&DecisionProcessAffected]，会议主席的决策过程受到影响，没有有效的措施可以阻止它的发生，要么选择去容忍它，或者重新设计业务流程，比如可以在评审意见总结后面增加一个任务：评审结果讨论，由一个评审委员会来做出最后的决定。

4．1．5．2 对基本安全目标取反建立反目标模型

当某些恶意目标难以直接鉴别出来，分析人员可以从基本安全目标的反目标着手，这种方法要求首先定义明确的、具体的、与应用领域相关的安全目标和安全属性，然而这不是一个简单的任务[Chivers2004]，因为每一个基本安全目标对不同的对象，甚至是同一对象在不同的上下文中的含义和要求是不同的。

对基本安全目标取反，建立反目标的方法和步骤如下：

1. 考察已建立的目标模型，鉴别安全敏感的对象，包括实体、关系、代理等，与有关人员讨论安全目标对这些对象的具体要求和相关的安全属性等等；
2. 对敏感对象特定的安全目标取反，得到初始的反目标；
3. 确定潜在的反目标的负责恶意代理。可以通过启发性问题(Heuristics)的提问和回答来鉴定，如“谁将从该反目标中获益？”等等。应注意，每个反目标可能对应几种类型的恶意代理。
4. 对于每一个反目标和每一种相应的恶意代理，鉴别它们的最终目的是什么，可通过“WHY”类型的问题的重复提问和回答来进行。不同的恶意代理其最终的目的可能是不一样的。
5. 采用同前一小节一样的方法，对反目标进行分析和处理，建立反目标模型和定义相应的管理策略。

例如对于评审者返回的评审意见，其完整性目标要求评审意见在传输和存储中均不能被改动，并且没有人冒充评审者返回评审意见，具体定义如下：

∀ pa: Paper, rvwr, rvwr' : Reviewer

rs=ReturnedReviewResult (pa, rvwr) ⇒

(ResultReturnedFrom (pa, rvwr) ∧ *SelectedReviewer (pa, rvwr')* ∧ *rvwr = rvwr')*

∧ ¬ *ResultAlteredInTransportation (rs)*

∧ ¬ *ResultModifiedInStore (rs)*

通过对上述基本的完整性目标取反，可以得到一个初始的反目标：

∃ pa: Paper, rvwr, rvwr' : Reviewer

rs=ReturnedReviewResult (pa, rvwr) ∧

(¬*(ResultReturnedFrom (pa, rvwr)* ∧ *SelectedReviewer (pa, rvwr')* ∧ *rvwr = rvwr')*

∨ *ResultAlteredInTransportation (rs)*

∨ *ResultModifiedInStore (rs))*

该反目标可以 OR－精化为三个子目标：

∃ pa: Paper, rvwr, rvwr' : Reviewer

rs=ReturnedReviewResult (pa, rvwr) ∧

$$(ResultReturnedFrom\ (rs, rvwr) \wedge SelectedReviewer\ (pa, rvwr') \wedge rvwr \neq rvwr') \qquad (anti\text{-}G_1)$$

$$rs=ReturnedReviewResult\ (pa, rvwr) \wedge ResultAlteredInTransportation\ (rs) \qquad (anti\text{-}G_2)$$

$$rs=ReturnedReviewResult\ (pa, rvwr) \wedge ResultModifiedInStore\ (rs) \qquad (anti\text{-}G_3)$$

对于第三个子目标 anti-G_3，可以鉴定出两个恶意代理：论文的作者和会议助理（会议主席也是一个潜在的恶意代理，但只能选择信任他，当然还有其它恶意代理，恶意代理的鉴别往往依靠分析员的经验）。恶意作者的最终目标可以很容易地识别出来，即使得他的论文得以通过，这一点在前面已经讨论过。对于会议助理的最终目标，通过对问题“为什么会议助理要修改评审意见？”的回答，可以鉴别出两个高层反目标：

Achieve [MakingPaperRejected]
Achieve [MakingPaperAccepted]

Achieve [MakingPaperRejected]指会议助理企图让一篇论文被拒绝，因为他不喜欢该论文的作者或其它原因；Achieve [MakingPaperAccepted]指会议助理企图要一篇论文得以通过，因为该论文作者是他的朋友或者其它原因。

识别出初始的反目标后，可以进行分析和精化，并根据终端目标定义管理策略，其方法和 4.1.5.1 中的方法一样，这里限于篇幅，不再重复讨论。

4．1．6 基于目标的管理需求分析方法小结

本文采用 KAOS 作为信息和系统管理需求分析方法的基础，主要是因为它已经经过较长时间的研究，并且被实践证明是切实有用的，许多成熟的支持工具也已经开发出来。只需要稍微的改动，完全适合用来进行信息和系统管理需求分析，在前面的介绍中也已经证明了这一点。

在这个阶段所定义的策略都是高层策略，它们只有经过进一步分析和翻译成低层可执行的策略，才能被系统所实施。正如在讨论策略的演化过程所介绍的那样，需要对所定义的策略进行各种必需的分析和验证，包括语法和语义验证、策略冲突和消解、策略的可行性验证等等。策略的冲突产生的原因是多种多样的，其中一个主要的原因是由于定义策略所根据的目标之间的冲突所导致的，对于这种类型的策略冲突的消解，既可以在目标层处理（目标冲突消解），也可以在策略层处理（策略冲突消解）。即使是在策略层处理，策略所依据的目标是分析员处理冲突的重要决策依据之一。

在这个阶段所定义的策略汇总后，还需有关人员进一步的讨论和筛选，因为可能存在大量重复和无用的策略。产生无用策略的主要原因是这些策略所要求达到的条件或采取的动作已经被系统实现，或者是客观事实，无需定义策略来重复要求。例如在前面已经讨论过的根据目标 G_{11} 所定义的职责策略要求论文提交系统通过 Email 向作者确认论文的提交，而事实上提交系统一般已经实现了这个功能，因此相应的职责策略是无用的，可以被删掉，但策略的删除必须小心，这主要是由于现在的信息系统的分布性和相互关联性，一个高层策略可能会被翻译成多个低层策略，并由多个系统来执行。例如根据同一目标 G_{11} 所定义的相应的授权策略则应该被保留，因为它不仅仅会影响到论文提交系统的配置，还可能影响到企业信息基础设施中其它系统的配置，如 smtp 服务器、防火墙等等。

事实上，这个阶段所定义的策略可以看作是策略模板，确定下来的策略的主体(subject)和对象(object)实质上是代理的类型（即角色）和对象的类型而不是具体的代理和对象，相当于在 RBAC 中仅仅确定了角色(role)和权限(permssion)的关系。在实际运行时，应根据业务流程的具体实例来实例化策略模板中的主体和对象。

4．2 采用用况和误用用况进行管理需求分析和策略制定

本节将简单介绍如何使用用况和误用用况进行信息和系统的管理需求分析以及指导管理策略的制定。

4．2．1 用况和误用用况

用况是需求工程中一个重要的分析工具，“用况可以用来定义系统或其它实体的行为，而无需揭示实体内部的结构。用况可以应用在任何的系统抽象层上。由于用况简单而又直观，它提供了描述系统功能概况的一个很好的方法”[OMG1999]。用况明确突出了参与者在系统中所扮演的角色，这一点使得它能够很容易地反映用户对系统的要求，并且为与用户的交流提供一个很好的平台[Jacobson1999]。现在，不仅在软件开发领域，在其它很多与交互和接口有关的系统开发和分析中，用况都得到了广泛的应用。

用况非常适合描述系统的功能需求，然而在信息和系统管理实践中，管理策略不仅要规定系统应该提供的功能、系统所允许到达的状态和发生的事件，还应该规定什么是系统不允许发生的和不能提供的。为此研究人员引入了误用用况，它是一种特殊类型的用况，定义了不希望系统/实体所发生的行为[Smdre2000]。相应

地，误用参与者被引入来代表实施误用用况的参与者。用况和误用用况可以在同一个用况图中表示，为区别它们，误用用况和误用参与者可用特殊的形式来表达，参见第二章的介绍。

用况之间的关系除了标准的关联，如“泛化”、“使用”、“包含”、“扩展”等等之外，为满足信息和系统管理需求的需要，还应增加一些新的（误用）用况之间的关联，如：

- “威胁(threatens)”：误用用况在某种程度上威胁了用况的安全；
- “阻碍(obstructs)”：误用用况在某种程度上阻碍了用况的实施；
- “阻止(prevents)”：用况所提供的功能可以在某种程度上阻止误用用况的实施；
- “监测(detects)”：用况所提供的功能可以在某种程度上监测到误用用况的发生。

4．2．2 用况和误用用况的确定

与策略的多层结构相对应，用况和误用用况的确定也可以多层次迭代进行，从企业的管理目标开始，一直到信息系统或更低的抽象层。低抽象层的用况可以将重点放在高抽象层的用况所没有考虑的具体细节或其它方面的要求，从而开始新一轮的分析[Sindre2000]。对于用况和误用用况的确定，许多该领域的学者提出了一些有用的建议[Alexander2003, Sindre2000]，主要包括以下几个步骤：

1．首先确定普通的用况，这些用况表示了系统所提供的服务、职员所执行的任务或管理活动等等；

2．引入误用用况，这些误用用况可能威胁或影响到现有的用况。有两种方法可用来识别误用用况：

- 考察知名的威胁或其它已知或曾经发生的威胁或事故等；
- 针对每个用况，进行启发式问题的提问和回答来识别误用用况，如“该用况会出错或被误用吗？”、“谁希望它会出错或误用它”、“他们如何误用它，或怎样能使它出错？”等等类似的问题；

3．确定已识别出的用况和误用用况之间可能存在的各种关系；

4．考虑如何阻止、处理或监测误用用况，因此可能需要增加新的用况。

新的用况又可能产生新的问题，因而又需要引入新的误用用况，所以可以说用况和误用用况的确定是一个多次迭代进行的过程，其结束条件是通过成本－效益分析，确定所采取措施的成本与威胁或障碍所产生的损失及其发生的可能性相比是否合算。

在本文的案例中，会议组织者通过 Web 向外部发布信息和提供论文提交接口，而网页的维护是一个必不可少的任务。会议的主页包含了与会议有关的重要信息，并且需要经常更新。主页的更新首先需要网页制作者准备新的网页，再由网站管理员负责网页的替换。这个看似简单的过程，其实包含了许多潜在的问题，下面将一步一步地按照上述的步骤来分析它：

1. 初始的用况有两个："网页制作"和"网页更新"，相应的参与者分别是网页制作者和网站管理员；

2. 对于"网页制作"，通过提问"网页制作会出错吗？"或类似的其它问题，可以知道网页制作者可能会有意或无意提供包含错误的网页，这些错误可能会导致会议组织者陷入尴尬的境况；对于"网页更新"，通过类似的分析可以知道，心怀不满的网站管理员可能会擅自将网页置换为带有恶意内容的网页。因此可以识别出两个新的误用用况："提供错误网页"和"错误网页更新"，这两个用况的误用参与者分别是网页制作者和网站管理员，他们同时也分别是"网页制作"和"网页更新"的参与者；

3. 确定用况之间的关系。很明显，"提供错误网页"包含"网页制作"，"错误网页更新"则包含并威胁到"网页更新"，另外"提供错误网页"将威胁到"网页更新"。

4. 为防止网页制作者提供错误网页，在制作者提交网页之后和由管理员上传之前，必须由某个信息主管对网页进行审核。同样，为监测错误的网页更新，任何网页更新发生以后，也必须由信息主管进行审核。

5. 只能信任和依赖信息主管，对于由于信息主管的工作疏忽或出于恶意而导致的错误，只能容忍它，否则所采用措施的成本太高，过于苛刻。

最后得到的用况和误用用况图如图 4－8 所示。

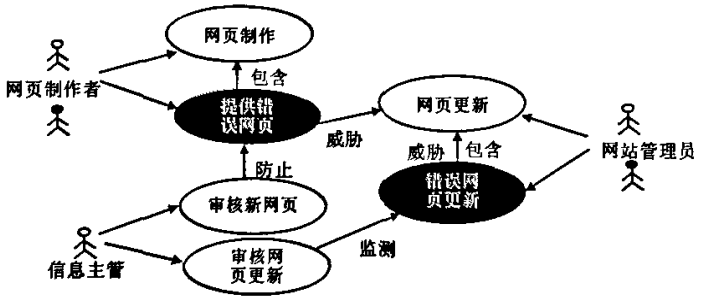


图 4－8 网页更新中的用况和误用用况

Fig. 4-8 Use and Misuse Case of Homepage Update

4．2．3 用况和误用用况模板(Template)

用况图只是对系统的一个框架性的表示，而“使用用况进行建模的本质在于它的文字描述部分”[Cockburn2001]。许多学者和研究机构提供了用况的文字描述的模板[Cockburn2001, Kulak2000, Kruchten2000]以及误用用况的模板[Sindre2001]。为适应信息和系统管理需求分析和制定管理策略的需要，本文在已有的模板的基础上，进行了适当的扩展和改进，确定的用况和误用用况的模板如下：

用况(误用用况)模板 Template of use (misuse) case:

- **标题 Title:** 每个用况（误用用况）都有一个唯一的、简明和直观的名字;
- **参与者（误用参与者）Actor (Potential Mis-actor)：** 对于误用用况，误用参与者有时是不确定的，因此所有潜在的参与者都应该列出来;
- **描述 Description:** 对整个用况（误用用况）简单概括性的描述;
- **前提条件 Preconditions:** 在用况（误用用况）初始化之前必须成立的条件;
- **主要场景 Main Scenario:** 列出为达到用况（误用用况）所要完成的目标参与者和系统要经过的步骤，这里描述的是正常的、没有意外发生时的步骤;
- **次要场景 Secondary Scenario:** 可供选择的其它的步骤;
- **例外（捕获点）Exceptions (Capture Points):** 对用况来说，指的是用况执行过程中可能会遇到的错误和非正常的情况，对于误用用况，该项给出了如何防止和监测误用的建议和选择[Sindre2001];
- **结束条件 Postconditions:** 在用况（误用用况）结束后必须要满足的条件;
- **触发条件 Trigger:** 触发用况（误用用况）的事件;
- **相关业务规则和控制原则 Related Business Rules and Control Principles:** 该企业或组织所采用的与本用况相关的业务规则和组织原则等;

- 关联 Relations (包含/扩展/泛化/阻止/监测/威胁/阻碍): 与其它用况（误用用况）的关系；
- 抽象层次 Abstraction Level: 该用况（误用用况）所在的抽象层；

表 4－1 和 4－2 分别给出了用况“网页更新”和误用用况“错误网页更新”的文字描述。

表 4－1 用况“网页更新”

Tab. 4-1 Use Case “Homepage Update”

- 标题: 网页更新
- 参与者: 网站管理员
- 描述: 网站管理员用新的经过信息主管审核过的主页替换原有的网页；
- 前提条件:

1. 新网页已准备好；
2. 网站管理员已准备好；

- 主要场景:

1. 网站管理员从指定的文件服务器上下载新的网页；
2. 网站管理员验证信息主管对新网页的签名；
3. 网站管理员从他的机器登录到 Web 服务器上的发布系统；
4. 网站管理员用新的经过信息主管审核过的主页替换原有的网页；
5. 网站管理员通知信息主管网页更新完成。

- 次要场景:

1. 网站管理员直接登录 Web 服务器更换网页。
2. 网站管理员目前未准备好，如请假未上班等，可以委派网络管理员代替他完成网页更新；

- 例外:

1. 网站管理员在新的网页还未经过主管审核好便将它下载下来；

- 结束条件:

1. 网页已成功地更换；

- 触发条件:
- 相关业务规则和控制原则:

1. 替换下来的网页应保存在历史网页数据库中，以备以后查询等。

- 关联 (包含/扩展/泛化/阻止/监测/威胁/阻碍):
- 抽象层: 企业业务层目标

表 4－2 误用用况“错误网页更新”

Tab. 4-2 Misuse Case “Wrong Homepage Update”

- 标题：错误网页更新
- 潜在的误用参与者：网站管理员
- 描述：网站管理员滥用职权，将错误的网页上传，使组织在经济上和名誉上受到损失；
- 前提条件：

  1. 网站管理员被授权更新网页；
- 主要场景：

  1. 当需要更新网页时，网站管理员改动或替换已经过信息主管审核的网页，并替换现有的网页；.
- 次要场景：

  1. 在无需更新网页或新网页还未经过主管审核时，网站管理员将错误的网页替换原来的网页；
- 捕获点：

1. 每一次网页更新都要被记录下来并且通知信息主管；
2. 当不需要更新网页时，网站管理员无权更新网页；

- 结束条件：

  1. 网页被错误的网页替换；
- 触发条件：
- 关联（包含/扩展/泛化/阻止/监测/威胁/阻碍）：包含、威胁 “网页更新”
- 抽象层：企业业务层目标

4．2．4 根据用况和误用用况制定管理策略

根据前面的介绍，一个信息和系统管理策略通常包括策略类型和四个基本部分：S、A、O 和 C。在策略的四个基本部分中，A（动作）是最重要的，因此可以采用“动作词汇识别法”来确定管理需求和制定管理策略，“动作词汇识别法”也是软件工程中系统需求分析的一个常用技术[Antón1997]。用况和误用用况文字定义中的所有动作词汇都可能对应一个或多个潜在的策略，这些动作词汇可能会出现在主要/次要场景、例外、捕获点、相关的业务规则和控制原则等。当动作词汇找到以后，对应的 S（主体）可通过对问题“谁或什么(WHO/WHAT)将负责实施该动作？”的回答来确定，而 O（对象）同样可通过对问题“什么或谁（WHAT/WHO）是该动作的对象？”的回答来确定。

接下来要确定策略的类型，这是一个具有挑战性的任务，需要分析员的经验和知识，需要对业务和信息系统的实现都有一定程度的了解。在第二章已经介绍过策略的分类，这里不再重复。策略分类的指导规则如下：

1. 如果动作要求访问或分配目标方的资源，或者会引起、引用目标方的操作的，则可能对应一个授权策略，常见的动作词汇有读、写、修改、访问、登录、上传、下载、启动、停止、使能等等。如果动作词汇是在用况的主要/次要场景、相关的业务规则和控制原则中发现的，或者是在误用用况中的捕获点或相关的业务规则和控制原则中发现的，则它对应一条潜在的（正态）授权策略。如果动作词汇前面有“必须”、“应该”、“立即”等强制性修饰，则还可能对应一个职责策略。相反，如果动作词汇出现在用况的例外或者误用用况的主要/次要场景中，则它可能对应一个禁止策略（负态授权策略）。

2. 对于用况，当定义了触发条件时，还可能对应一条或多条职责策略，要求启动主要场景中的第一个步骤，或者在某些条件成立时，启动各次要场景中的第一个步骤。对于误用用况，当定义了触发条件时，对应一条或多条职责策略，要求采取一定的措施阻止、监视、记录误用用况的发生。

3. 如出现“委托”等类似的词汇，则可能对应一个授权策略。

管理策略的约束部分可通过以下步骤来确定：

1. 寻找时间和地点状语连接词，如“在...之前”、“在...之后”、“当...时”、“在...期间”、“在...位置”、“在...内部”等等，这些状语限制了策略的有效期间和地点。

2. 寻找其它对执行动作的限制声明，如质或量的要求和限制、主体/对象/环境等的状态要求和限制等。

3. 用况（误用用况）前提条件中定义的条件是所有从该用况得出的策略的可能的约束条件。

尽管上述的方法可以指导分析人员根据用况（误用用况）制定管理策略，但最后所制定的策略的质量好坏仍主要依靠分析员和策略制定者的经验和知识。因此策略的制定过程将是一个反复进行的过程，并要求企业管理者和系统专家的积极参与。表 4—3 和 4—4 分别列出了根据“网页更新”和“错误网页更新”所确定的管理策略。策略的形式与 4.1 节相同，策略的编号表明了策略具体根据用况或误用用况的哪一部分确定的，如 m1 表明是根据用况（误用用况）的主要场景的步骤 1 确定的，s1 则是根据次要场景的步骤 1 确定的。

表 4－3 根据用况“网页更新”定义的管理策略

Tab. 4-3 Polices Corresponding to Use Case “ Homepage Update”

<table>
<tr><th>管理策略</th><th>管理策略</th></tr>
<tr><td>m1: auth+ download Homepage {
subject /Website Administrator;
target /FileServerX/newHompage;
action download();
}
网站管理员被授权从文件服务器 X 上下载新的网页。</td><td>m2: oblig verifySignature {
On homepageDownloaded(newpage)
subject /Website Administrator;
do verifySignature (newpage)
}
网站管理员必须验证信息主管对新网页的签名。</td></tr>
<tr><td>m3: auth+ logonPublishingSys {
subject /Website Administrator;
target /WebServer/PublishingSystem;
action remoteLogon();
}
网站管理员被授权从他的控制台登录到网站的发布系统。</td><td>m4: auth+ replace Homepage {
subject /Website Administrator;
target /PublishingSys/Hompage;
action replace();
}
网站管理员被授权更新网页。</td></tr>
<tr><td>m5: oblig notifyHomepageUpdate {
on homepageUpdated(newPage);
subject /Website Administrator;
do notifyHomepageUpdate (InfoOfficer, newPage),
}
网站管理员在网页更新后，必须立即通知信息主管。</td><td>s1: auth+ logonWebServer {
subject /Website Administrator;
target /WebServer;
action logon();
}
网站管理员被授权可以直接登录到 Web 服务器。</td></tr>
<tr><td>s2: deleg delegHomepageReplace {
grantee /Network Administrator
subject /Website Administrator;
target t1=/WebServer+ t2=/Webserver/Homepage;
action t1.logon(), t2.replace();
}
网站管理员可以委托网络管理员来更新网页。</td><td>e1: auth- dlHomepageBeforeApproval {
subject /Website Administrator;
target t= / FileServerX/newHompage;
action download();
when t.state=”unapproved”
}
在新网页未审核之前，禁止网站管理员从文件服务器上下载它。</td></tr>
<tr><td colspan="2">r1: oblig logHomepageUpdate {
on homepageUpdated(oldpage, newpage);
subject s= /WebServer/publishingSys;
do logHomepageUpdate (oldpage, newpage);
}
发布系统应记录每次网页更新事件。</td></tr>
</table>

表 4－3 根据误用用况“错误网页更新”定义的管理策略

Tab. 4-3 Policies Corresponding to Misuse Case “Wrong Homepage Update”

管理策略	管理策略

m1: auth- download Homepage {
subject /Website Administrator;
target /FileServerX/newHompage;
action modify();
}
禁止网站管理员修改存储在文件服务器 X 中的新网页。
	s1: auth- alter Homepage {
	subject /Website Administrator;
	target /PublishingSys/Homepage;
	action replace(), alter();
	when /FileServerX /newHompage state=" empty" || "unapproved"
	}
	禁止网站管理员在不需要网页更新的时候，去更新网页。

c1: oblig logHomepagechange {
on homepagechanged(userid, oldpage, newpage);
subject s= /WebServer/publishingSys;
do s.log(userid)→s.store(oldpage)→s.notify(InfoOfficer, newpage);
}
Web 服务器的发布系统必须记录网页的每次更新，并及时通知信息主管。	

4．3 小结

编写和制定明确、全面、精确和完整的系统管理策略对于成功的企业信息资源的管理是至关重要的。而目前随机的、面向设备和技术的管理策略的编写方法已经不能满足需要，在制定信息资源管理策略时，需要有一个系统的方法，而这个方法的起点应该是分析和确定信息资源管理的要求。在软件开发过程中，系统需求分析是一个重要的、不可或缺的环节，已得到充分的重视。然而在信息和系统的管理中，专家和学者们往往把研究重点放在了策略的说明、分析细化、翻译和发布实施机制上，却忽视了一个重要的环节，即这些策略从哪里来，其目的是什么等，这就需要一个将企业的业务目标映射到信息和系统管理目标的环节和机制，本章首次明确强调了这一环节的重要性，并给出了相应的解决方案。

Moffett 曾建议可以将高层策略看作是一种系统需求，而低层策略则是需求的一种实现，因此系统需求分析里的许多概念和技术可以在这里被借鉴和采用。但需要进行适当的改进以适应信息资源管理的需求分析。

基于目标的需求分析方法可以涵盖系统的功能需求和非功能需求，并将目标和需求通过精化过程联系起来，是目前公认的最具潜力的一种需求分析技术。因

此，本文选择了基于目标的需求分析方法作为信息和系统管理需求分析和编写管理策略的一个主要方法。本章给出了基于目标的企业信息和系统管理需求分析模型和框架，该模型以 KAOS 需求分析模型和方法为基础，为应用于信息和系统管理需求分析而进行了相应的改进和扩展，KAOS 的一个主要优点是使用时序逻辑规则来表达目标，因此能够使用形式化的方法对目标进行分析、细化和验证；本章还给出了建立目标模型的方法和步骤，以及根据目标模型确定管理需求和制定管理策略的方法和规则；在信息资源的安全管理方面，本章给出了如何建立反目标模型和根据反目标模型确定管理策略的方法。

其它的需求分析技术通过适当的扩展和改进，同样也可以用来分析信息资源管理需求。本章简单探讨了用况和误用用况在信息和系统管理需求分析中的应用，提供了相应的确定用况和误用用况的方法，给出了适用于信息和系统管理需求分析领域的用况和误用用况模板，以及根据用况和误用用况制定信息资源管理策略的方法。

第五章 策略的分析和验证

在策略演化过程中的各个阶段，对应于策略的各个抽象层次内部和各层次之间，都需要对策略进行各种分析和验证，除包括常见的策略的语法、语义分析，关联分析，冲突的检测和消解外，还包括策略的完整性、完全性和可行性分析，以及策略的优化、合并、分解等等。由于在策略的演化过程中各个阶段所处的上下文环境不同，策略的抽象层次不同，因此，各个阶段的分析和验证的侧重点和内容也不一样，所采用的分析方法和建模工具也不尽相同。

策略的分析和验证是目前该领域的薄弱环节，也是阻碍基于策略的管理方法得到广泛应用的一个主要原因之一[Bandara2005]。

由于所涉及的策略分析验证种类繁多，并且依赖于具体的应用领域，无法也没有必要在一篇文章中全部予以讨论，本章就本文的实例，分别在策略的高层、中间层和低层三个不同抽象层中各选出一到两项重要的、目前该领域中尚未成熟、亟待解决的一些策略分析和验证内容进行讨论，并给出本文的方法。

5.1 业务流程实例的授权管理的建模与分析

在 4.1 节中，介绍了如何根据对业务流程中各项任务的业务目标所作的分析来制定信息和系统管理策略，并且指出所定义的策略实质上是策略的模板，其中所确定的策略的主体(subject)和对象(object)是代理和对象的类型而不是具体的代理和对象。只有在业务流程的某个具体实例运行时，当各个具体的代理和对象确定下来后，对各个策略模板进行实例化，最后才能得到各个策略实例。在策略的实例化过程中，可能会遇到下列问题需要解决：

1. 如何实现策略实例层次上的安全约束。例如，职责分离准则 SoD(Separation of Duties)就是其中一个重要的安全约束，该准则由著名的 Clark & Wilson 模型提出，其主要目标是防止欺诈(Fraud)，欺诈是许多应用领域中所面临的主要威胁之一。为降低欺诈的风险，SoD 准则要求一个关键活动被分解为几个子部分，每个子部分被视为是相互冲突的而且必须由不同的人员来执行，因此，除非这几个人合谋，否则欺诈行为难以实施。职责分离准则 SoD 被进一步分类为静态 SoD 和动态 SoD。在业务流程中，如果指派合适的角色给每个任务，并确保相互冲突的任务由不同的角色来执行，这

样就能实现静态 SoD。相反，动态 SoD 只能在系统运行时才能实现，它要求在特定的业务流程实例运行时，为每个任务指派合适的执行者，并确保相互冲突的任务不能由同一个人或相互冲突的人员来执行。动态 SoD 是许多研究的重点，如未特别说明，在下文中，SoD 即指的是动态 SoD。其它的动态约束根据应用领域有很多种，如要求某个任务的执行者必须是另外一个任务执行者的部门主管，病人的病例只能有他的负责医生来编写等等。这些约束只有在策略实例层上才能得以实现。

2. 为确保业务流程的顺利进行，合理的人员安排和资源分配是十分重要的。对业务流程的研究早已是一个热门课题，然而大多数的学者的研究重点是过程控制，而忽视了资源规划。这里的资源既包含信息、设施等资源，也包含人力资源。在进行资源特别是人力资源规划时，应如何实现安全和管理约束，如何处理意外等等。

3. 根据业务目标分析而制定的策略，只反映了策略间的纵向关联，而策略之间还因为任务之间的依赖关系而存在着横向关联。一个业务流程中的任务因各种依赖关系而组成一个完整的过程，这些依赖关系包括时态、语义和原子关系等等[Thomas1994]。那么，根据各个任务而确定的管理策略也必然存在着相应的关联。在验证策略、进行资源规划和实例化策略时，应如何将这种关联因素考虑进去。

业务流程中的每一个任务对应了一组策略模板，这些策略模板定义了与执行任务相关的权限和职责，当业务流程中的各个任务的执行者和所访问的资源确定下来后，即可以对这些模板实例化，得到一组组策略实例。因此在分析和验证策略时，为简化起见，可以以任务为单位，从而可以降低分析的复杂度。

下面将介绍如何使用彩色 Petri 网对业务流程、安全约束和人力资源进行建模，从而分析和验证业务流程的权限管理，并进行人力资源规划。

5．1．1 分析假设条件

本节仍以案例分析中的论文收集和评审过程为例，对业务流程授权和职责进行建模与分析。由于论文作者的不确定性，并且是一个外部角色，这里不对论文提交任务做具体的分析，只是把它作为启动一个业务流程实例的事件。

会议助理，评审者和会议主席都是指派给相关任务的角色，根据组织结构定义建立的角色层次结构如图 5－1 所示。角色层次结构定义了角色的偏序关系，支配角色的用户被允许扮演被支配的角色去执行相应的任务。

下面给出了各个角色相应的用户名单，以用于稍后对模型的分析。

会议助理：Alice；评审者：Bob, Carl, Dick；会议主席：Eric。

同时，有关的安全约束定义如下，它们主要是动态安全约束：

c1：论文的收集者不能再评审该论文；

c2：一篇论文必须由两个不同的评审者评审；

c3：论文评审意见的总结者不能是该论文的评审者；

c4：评审结果必须由该论文的收集者反馈给作者。

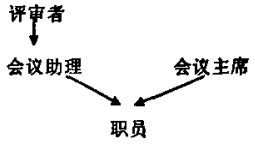


图 5－1 角色层次结构

Fig. 5-1 Role Hierachy

5．1．2 基于彩色 Petri 网的业务流程授权模型

使用 Petri 网建模，不仅可以描述系统的特性，还提供了对系统进行分析的技术和方法。下面首先介绍基本的 Petri 网和它的一个扩展，即彩色 Petri 网，然后给出使用彩色 Petri 网来建立业务流程授权模型的方法。

5.1.2.1 彩色 Petri 网 CPN (Colored Petri Net)

Petri 网是一个有向图，包含两种结点：位置(Place)和转换(Transition)结点。不同类型结点之间可由有向弧连接。在 Petri 网的图形表示法中，位置结点一般由圆圈表示，转换结点由竖杠表示。

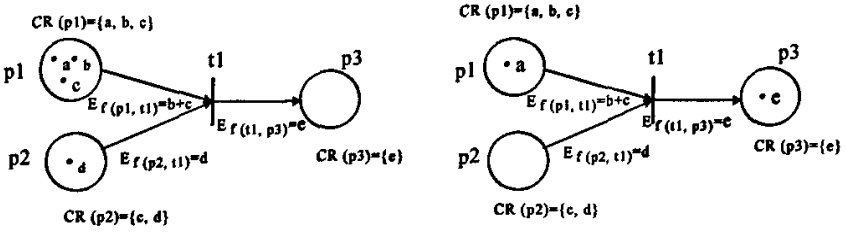
一个转换结点 t 是一个位置结点 p 的输入转换，当且仅当存在一条从 t 到 p 的有向弧；一个转换结点 t 是一个位置结点 p 的输出转换，当且仅当存在一条从 p 到 t 的有向弧。转换结点 t 的输入和输出位置结点具有类似的含义。

一个位置结点包含零个或多个令牌(Token)，一个位置结点中令牌的数量称为该位置的状态。Petri 网的状态则是所有位置的状态的集合。令牌的数目是变化着的，转换结点是 Petri 网中的活跃因子，它根据激发规则来改变 Petri 网的状态。

从一个状态 M_1 出发，如果存在一个转换激发序列使得状态从 M_1 转变到 M_n，则称状态 M_n 是从 M_1 可达的。可达性是一个十分有用的特性，可以用来分析系统的动态特征，缺点是其复杂度随着结点数量的增加呈指数级增长。

高层 Petri 网是基本 Petri 网的扩展，可以提供更强的表达能力，它从较高的抽象层次为系统建立模型，从而减小了模型的复杂性。彩色(colored)、时序(timed)和层次(hierarchy) Petri 网是三种广泛使用的高层 Petri 网。

在经典的彩色 Petri 网(CPN,Colored Petri Net)中，每个令牌定义了一个属性，称为“颜色”，每个 CPN 定义了一个颜色集。每个位置结点也定义了一个颜色集，是 CPN 颜色集的子集，定义了该结点的合法颜色。一个令牌只有在它的“颜色”是某个位置结点的合法颜色时，才能被允许进入该位置结点，在许多彩色 Petri 网的扩展或变体中，这一点不再是必须的，可以重新定义。转换结点根据所消耗的令牌的“颜色”和激发规则来决定所产生的令牌的“颜色”。一个转换结点 t 只有在它的输入位置结点 p_i 拥有的某种颜色的令牌数大于或等于相应的弧 f(p, t) 所对应的弧函数 $E_{f(p,\,t)}$ 规定的数量时，它才是可激发的。这些令牌在转换激发时被消耗，并且产生了弧函数 $E_{f(t,\,Pi+1)}$ 所定义的相应数量和颜色的令牌到相应的输出位置结点 p_{i+1}[Jensen1992]。



(a) t1 激发前　　(b) t1 激发后

图 5－2 一个简单的彩色 Petri 网

Fig. 5-2 A Simple Color Petri Net

图 5－2 是一个简单的彩色 Petri 网，包含三个位置结点 P1、P2、P3 和一个转换结点 t1。P1、P2 和 P3 的颜色集分别为 CR (P1)={a, b, c}、CR (P2)={c, d}和 CR (P3)={e}，每个弧的弧函数在图中标出。图 5－2(a)所示的是 Petri 网的初始状态，可以看出此时 t1 是可激发的。图 5－2(b)所示的是 t1 激发后的状态，P1 中颜色分别为 b 和 c 的两个令牌，和 P2 中颜色为 d 的一个令牌被消耗，同时产生一个颜色为 c 的令牌并放入到 P3 中。

5．1．2．2 业务流程授权管理的建模方法

每一个业务流程都包含三个流：控制流（控制着各项任务的进程），数据流和授权流。为保证业务流程的顺利进行，数据流和授权流应与控制流同步，因此可以认为控制流支配着数据流和授权流。控制流涉及任务间的各种依赖关系，如时态，语义和原子等依赖关系。数据流涉及数据的产生、消耗及其在任务之间的流动，数据在任务的执行过程中被创建，访问和删除，可以看作是控制流的辅助流，因而，当一个业务流程实例化后，相应的数据流也已经是可确定的，这就是为什么在下面的授权模型中，权限和职责的定义和分析仅需要定义角色或用户以任务为单位的权限和职责，用户因执行任务的需要才被授予资源访问权限和规定相应的职责，表现在对各任务所对应的策略模板进行具体的实例化。这种建立在高抽象层次上的权限和职责管理降低了分析的复杂度，使管理和分析人员可以专注于业务流程实例化中人员安排的合理性分析和对各种安全和管理约束的实现上，这也是在该策略层次上，策略定义和分析的一个主要任务。

使用 Petri 网建立业务流程的控制流模型是十分直截了当的，其中，任务被映射为位置结点，任务切换控制条件映射为转换结点。论文收集和评审过程的 Petri 网模型如图 5－3 所示。由于安全约束 c2 要求一篇论文必须由两个评审者评审，因此论文评审可以看作是两个并行的任务。此外，为了标识业务流程的结束，在最后增加了一个位置结点 P_{finish}，而业务流则由论文提交事件来启动。

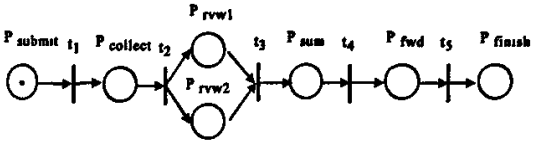


图 5－3 论文收集评审过程的 Petri 网模型

Fig. 5-3 The Petri Net Model of the Paper Collection and Review Process

假设业务流程中，一个任务只能有一个用户负责执行。复杂的、需要多个用户共同负责的任务，经过仔细分解后，可以满足上述要求。当然，一个任务可能由多个活动和步骤组成，这些活动可能由负责任务的用户以外的用户来实施，但如果各任务的负责人确定下来后，这些活动的实施者也可以相应地确定下来，如果是随机的，无法由前面的方法确定的，可将该任务分解成多个子任务，因此在这里的分析中，只考虑以任务为单位的人员安排。

基于 CPN 的业务流程授权模型将控制流和授权流结合一起。业务流程的授权和职责中各相关元素和安全约束采用以下方法映射到 CPN：

(1) 角色 r 映射到位置结点 p_r，称为角色位置结点，它定义有一个颜色集，包含了所有合法的颜色，即合法的用户。如果把 p_r 定义为某个转换 t_S 的输入结点，而 t_S 又是代表某个任务的位置结点 p_T 的输入转换，则表示了该任务 T 和角色 r 间的指派关系。角色的层次结构关系可以通过在支配角色和被支配角色之间加入一个辅助转换结点来表示，这样支配角色位置结点中的令牌可以被传递到被支配角色位置结点去，从而表示一个支配角色的用户可以扮演被支配角色。

(2) 一个用户由一个特定颜色的令牌表示。安排用户为某个角色则表示成代表该用户的令牌被放置到代表相应角色的位置结点中。这里假设在同一个业务流程实例中，每个用户只能扮演一个角色，这一点类似于在 RBAC(基于角色的访问控制)中，规定一个用户在一个会话期只能扮演一个角色[Sandu1996]。

(3) 授权一个用户去执行某个任务 T 表示为代表该用户的令牌被放置到代表该任务的位置结点 p_T 中。

(4) 执行任务 T 所需的相关的权限在 t_S (p_T 的某个输入转换) 激发时才被授予其执行者，并且在 t_E (p_T 的某个输出转换) 激发时被取消。t_S 激发时将一个代表某个用户的令牌放置到 p_T 中，而 t_E 激发时，则从 p_T 中消耗掉该令牌，该令牌在 p_T 中逗留的时期表示该任务的执行状态和执行时间以及该用户拥有执行任务所需权限的时间段。在这里，一个 p_{Ti} 的输出转换 t_E 也可能同时是另一个 p_{Tj} 的输入转换。

(5) 授权流与业务流程的同步是由以下的方式实现的。由于业务流程授权模型将控制流和授权流集于一体，其中，每一个任务的输入转换结点 t_S 拥有两种类型的输入位置结点，一种是控制流中的输入结点，另一种是授权流中代表角色的位置结点 p_r。这样，除了满足控制流的要求外，转换结点 t_S 只有在相关角色的用户是可用的情况下，才是可激发的。如果从授权模型中去掉代表角色的位置结点和相关的弧，剩下的将是一个完整的代表某个业务流程的业务流程模型，因此，如果业务流程模型已经存在的话，授权模型可以直接建立其上。

(6) 职责分离 SoD 约束可根据不同的情况采用不同的方法来建模.

- 相互冲突的任务需要由不同角色的用户来执行

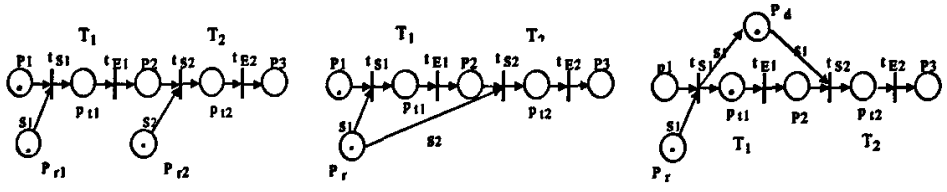
由于假设在同一个业务流程实例中，一个用户只能安排一个角色，所以相互冲突的任务的输入转换 t_S 如果定义了不同的角色位置结点作为输入位置结点，它们将从不同的用户集中选择执行者，从而实现了 SoD，参见图 5-4(a)。

- 相互冲突的任务需要相同角色的用户来执行

相互冲突的任务的输入转换 t_S 如果定义了相同的角色位置结点作为输入位置结点，它们将从相同的用户集中选择执行者。如果一个用户被选择去执行某个任务，则代表该用户的令牌被消耗，从而对其它任务来说将是不可用的，其它任务只能从该角色剩下的用户中选择执行者，这样 SoD 得以实现，参见图 5-4(b)。

- 绑定任务 (在[Atluri1996]中称作断言型 SoD)

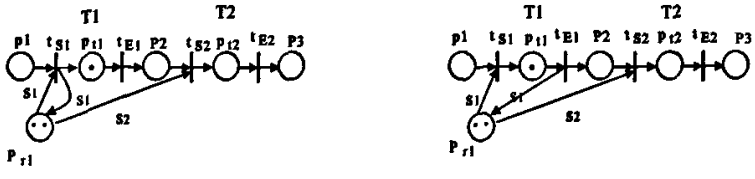
这种类型的约束要求一个用户如果执行了任务 T_i，则他还必须执行同一个业务流程中另外一个任务 T_j。为表示这种约束，需增加一个新的角色位置结点定义为任务 T_i 的输出结点以及 T_j 的输入结点。当 T_i 的 t_S 激发时，将产生与消耗的令牌颜色相同的令牌到 p_d 中，且该令牌则是 T_j 的唯一选择，如图 5-4(c)所示。



(a) 相互冲突的任务需要不同角色的执行者　(b) 相互冲突的任务需要相同角色的执行者　(c) 绑定任务

图 5－4 SoD 的建模方法

Fig. 5-4 Modeling of SoD



(a) 令牌由 t_S 产生　(b) 令牌由 t_E 产生

图 5－5 相互不冲突的任务

Fig. 5-5 Non-Conflict Tasks

(7) 同一个用户可允许执行多个相互不冲突的任务，这种情况可以表示为 t_S 激发时，在从角色位置结点消耗一个令牌时，再产生出同一颜色的令牌到某个角色

位置中去，如图 5－5(a)，(b)所示，令牌是由 t_S 还是由 t_E 产生取决于是否允许用户同时执行多个任务。更复杂的情况同样可以使用这些基本方法进行建模，图 5－6(a)-(d)中给出了一些范例。

注意在图 5－6(a)-(c)中，加入了辅助转换结点 t_a，这些转换结点属于授权流，其优先级低于控制流中的转换结点，就是说尽管它们是可激发的，也只有在需要的时候才会被激发，也即当需要安排用户负责执行任务时，才去激发它们。

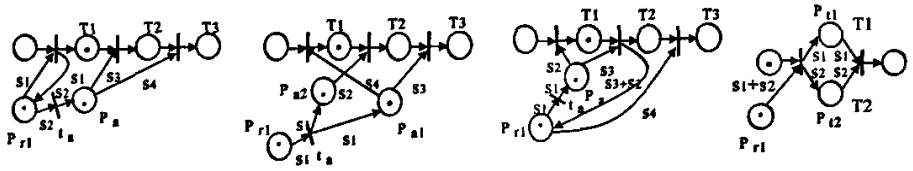


图 5－6 复杂的情况

Fig. 5-6 Complex Situation

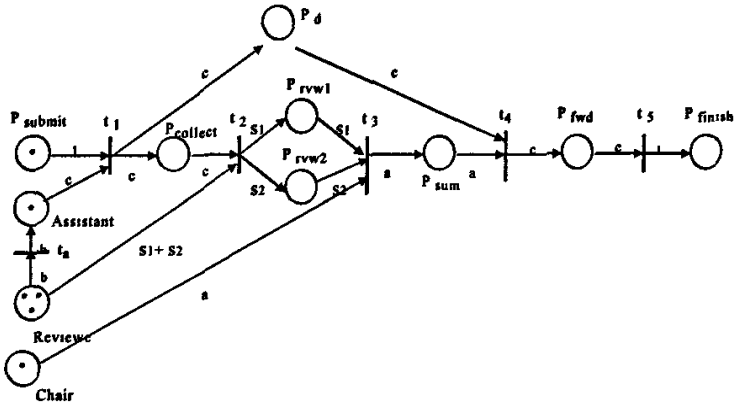


图 5－7. 基于 CPN 的论文评审过程的工作流的授权模型

Fig. 5-7 CPN-based Paper Review Workflow Authorization Model

图 5-7 是会议论文评审业务流程的 CPN 授权模型。位置结点 Assistant、Reviewer 和 Chair 对应于三个相应的角色。角色结点 Reviewer 和 Assistant 之间的转换结点 t_a 表明一个评审者可以扮演会议助理的角色。$P_{collect}$、P_{rvw1}、P_{rvw2}、P_{sum} 和 P_{fwd} 对应于工作流中相应的任务。整个业务流程实例由论文提交事件启动，并初始化。在 Petri 网的初始状态中，位置结点 Assistant、Reviewer 和 Chair 被放入代表各相应成员的令牌，这是根据组织的结构和人力资源管理信息而确定的。因为组织中的人员是随时变化着的，所以这些信息是动态的，不同的业务流

程实例可以选择执行者的用户集可能是不相同的。从图中可以看出，如果一个评审者扮演了会议助理的角色去收集论文，那么他将不可能再评审该论文，从而满足了安全约束 C1。同样，弧 f (Reviewer，t_2)要求两个不同的令牌 s1 和 s2 来执行评审工作，从而满足了约束 C2。C3 的满足非常直接，即只有 Chair 被定义为 t3 的输入角色位置结点。在 t1 和 t4 之间增加一个辅助角色位置结点 P_d，并规定弧 f(t1，P_d)、f(P_d，t4)和 f(Assistant，t1)有着相同的弧函数 c，即相同的令牌被消耗和产生，从而实现了约束 C4。

5．1．3 业务流程的授权模型的分析

使用 Petri 网进行系统建模的一个优势是它得到了许多分析技术的支持，这些技术如线性代数，可达树，覆盖图，和模型检验技术等等可以用来验证系统的性质和计算系统的性能。下面分别介绍如何使用线性代数技术来验证授权模型的可达授权状态和 Petri 网的覆盖图来计算业务流程的有效执行链。应该指出，覆盖图同样可以用来验证可达授权状态。

5．1．3．1 业务流程的可达授权状态

在 CPN 授权模型中，授予用户权限被表示成将代表该用户的令牌放置到代表某个任务执行状态的位置结点中去，因此业务流程的可达授权状态问题等价于 Petri 网中的可达性分析，即分析是否可以从 Petri 网的初始标志状态 M_0 经过某个转换激发序列到达代表某个授权状态的标志状态 M。

线性代数技术可以用来进行 Petri 网的可达性分析。在这种方法中，Petri 网的状态 M 表示为一个 m x 1 阶矩阵，其中 m 是位置结点的数量，从初始状态 M_0 出发，状态 M 是可达的当且仅当 U 存在一个非负的整数解，满足:

$$M=M_0+AU$$

其中，A 是关联矩阵，表示由于转换激发所导致的状态转变。A 可以从 Petri 网定义中获得: A 是一个 m x n 阶矩阵，m 和 n 分别是位置和转换结点的数量，$a_{ij}=a_{ij}^{+}-a_{ij}^{-}$，$a_{ij}^{+}(a_{ij}^{-})$ 对应于从转换结点 t_j 到输出 (输入) 结点 p_i 的弧的函数定义的所产生 (消耗) 的令牌数 [Murata 1989]。

这种方法要求所分析的 Petri 网必须是非循环的，即在 Petri 网中，不存在可循环执行的路径。 Alturi 和 Huang 指出并证明了如果一个过程的定义是一致性（consistent）的，那么它所对应的 Petri 网是非循环的[Atluri1996&2000]。在前面曾提

到，授权模型可被视作是由控制流和授权流两个子网组合而成，代表某个任务的转换结点均包含两种类型的输入位置结点：控制和角色位置结点。转换结点只有在控制条件和角色/人员条件都满足的情况下才可以被激发。如果控制流是非循环的，则最终的授权模型也必将是非循环的。因此，尽管看上去图 5－5、5－6 中有回路存在，但它们不会引起相关转换的循环激发，仍可以视作是非循环的。

图 5-8(a)给出了图 5-7 所示的工作流授权模型的关联矩阵 A，初始状态中，会议助理结点中放入代表 Alice 的令牌，评审者结点中放入代表 Bob，Carl 和 Dick 的令牌，会议主席结点中含有代表 Eric 的令牌。初始状态如图 5－8(b)所示。

$$A=\begin{array}{c} \\ P\ start \\ Assistant \\ Reviewer \\ Chair \\ P\ collect \\ P\ rvw1 \\ P\ rvw2 \\ P\ d \\ P\ sum \\ P\ fwd \\ P\ finish \end{array}\begin{array}{c} \begin{array}{cccccc} t_a & t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ b & -c & 0 & 0 & 0 & 0 \\ -b & 0 & -(s1+s2) & 0 & 0 & 0 \\ 0 & 0 & 0 & -a & 0 & 0 \\ 0 & c & -c & 0 & 0 & 0 \\ 0 & 0 & s1 & -s1 & 0 & 0 \\ 0 & 0 & s2 & -s2 & 0 & 0 \\ 0 & c & 0 & 0 & -c & 0 \\ 0 & 0 & 0 & a & -a & 0 \\ 0 & 0 & 0 & 0 & c & -c \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \quad M_0=\begin{bmatrix} 1 \\ Alice \\ Bob+Carl+Dick \\ Eric \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad M_1=\begin{bmatrix} - \\ - \\ - \\ - \\ - \\ - \\ - \\ - \\ - \\ Dick \\ - \end{bmatrix} \quad AU=\begin{bmatrix} -1 \\ b-c \\ -b-s1-s2 \\ -a \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ c \\ 0 \end{bmatrix} \quad M'=\begin{bmatrix} 0 \\ Alice \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ Dick \\ 0 \end{bmatrix}$$

(a) 图 5 中 CPN 模型的关联矩阵 A　(b) M_0　(c) M_1　(d) AU　(e) M'

图 5－8 解方程 $M=M_0+AU$

Fig. 5-8 Equation $M=M_0+AU$

如果要验证 Dick 是否会被授权执行论文评审结果反馈的任务，则需做验证图 5－8(c)所示的标志状态 M_1 的可达性，即代表 Dick 的令牌是否能被放置到位置结点 p_{fwd} 中，验证过程即是求方程 $M_1=M_0+AU$ 的非负整数解。这里可以很容易地得到一个 U 的非负解，$U=[1\ 1\ 1\ 1\ 1\ 0]^T$。如果令 b=c=Dick，s1=Bob，s2=Carl，a=Eric，则方程的右边等于 M'，如图 5－8(e)所示。可以看出 M'与 M_1 相匹配，即它是方程的一个解，因此 Dick 能够被授权执行返回论文评审结果的任务。

5．1．3．2 业务流程的有效执行链

业务流程的一个执行链是指该业务流程中所有相关任务的一个可能的人员安排，在该执行人员的安排下，一个业务流程实例可以通过相关任务的执行而得以

完成。如果任务和执行人员的分配不违反有关的安全约束和管理要求，则相应的执行链称为有效执行链[Knorr2001]。

对于业务流程，无论是设计还是实例执行，有效执行链的计算和分析都是必需的。对于业务流程的设计，如果无法找到相应的有效执行链，管理方应考虑重新设计业务流程或组织结构，否则，为了业务流程能够得以执行，就要降低安全约束的要求。对于在业务流程实例的执行期间，有效执行链的计算同样也很重要，因为许多动态安全约束只有在实例执行时才得以贯彻，而业务流程的执行期间可能会产生人员的变动，如职员职务的变动或离职等等，使得原先的安排不再合适，需要及时重新计算有效执行链等等。执行链的变化，也决定了相应的授权和职责的变化，这些变化需要及时反映到有关的管理策略和系统配置上。

Petri 网的覆盖图可以用来计算业务流程的有效执行链。图 5－9 给出了图 5－7 所示的业务流程的授权模型的覆盖图，图中的每个节点对应于该 Petri 网的一个可能的状态。每个节点包括两行，第一行有七项，分别代表七个控制（任务）位置结点的标识，即 P_{submit}、$P_{collect}$、P_{rvw1}、P_{rvw2}、P_{sum}、P_{fwd}、和 P_{finish}；第二行有四项，分别代表四个角色位置结点的标识，即 Assistant、Reviewer、Chair、和 P_d。如节点 1 代表了模型的初始状态，第一行为(1000000)，表示 P_{submit} 有一个令牌，其余六个控制位置结点为空；第二行为(A, B+C+D, E, 0)，表示角色位置结点 Assistant 有一个令牌 A，Reviewer 有三个令牌 B、C 和 D，Chair 有一个令牌 E，P_d 没有令牌（这里 A、B、C、D 和 E 分别代表 Alice、Bob、Carl、 Dick 和 Eric，这样做仅仅是为了表示上的方便）。覆盖图中节点间的每个箭头代表一次转换的激发，例如节点 1 和 2 之间的箭头标有 t1，表时在节点 1 代表的初始状态时，t1 的激发将 Petri 网的状态改变成节点 2 所代表的状态。在节点 2 中，第一行为(0A00000)，表示令牌 A 被放入 $p_{collect}$，代表 Alice 被授权执行收集和送审论文的任务。由于在彩色 Petri 网中，令牌之间因颜色的不同而是有区别的，所以在同一状态同一转换的激发，由于所消耗的令牌不同，可能会导致不同的结果状态，例如在初始状态 1，转换 t_a 也是可激发的，t_a 的激发根据所消耗的令牌的不同，可能导致三个结果，分别由节点 3、4 和 5 表示。

通过分析图 5－9，可以得出以下几点结论：

- 从根到叶子节点的一个路径就是一个有效执行链，并给出了任务的执行顺序和相应的执行人员；

- 两个或多个路径可能对应一条有效执行链，如路径 1-2-7-16-25-34、路径 1-2-9-18-27-38-42 和路径 1-3-9-18-27-36-42 对应于一个有效执行链，这是由于增加了辅助转换节点 t_a 导致的；
- 从图 5－9 中总共可以找出 6 条有效执行链：

(1). (Alice, 论文收集并送审)-(Bob, 论文评审)-(Carl, 论文评审)-(Eric, 评审意见总结)-(Alice, 评审结果反馈)；

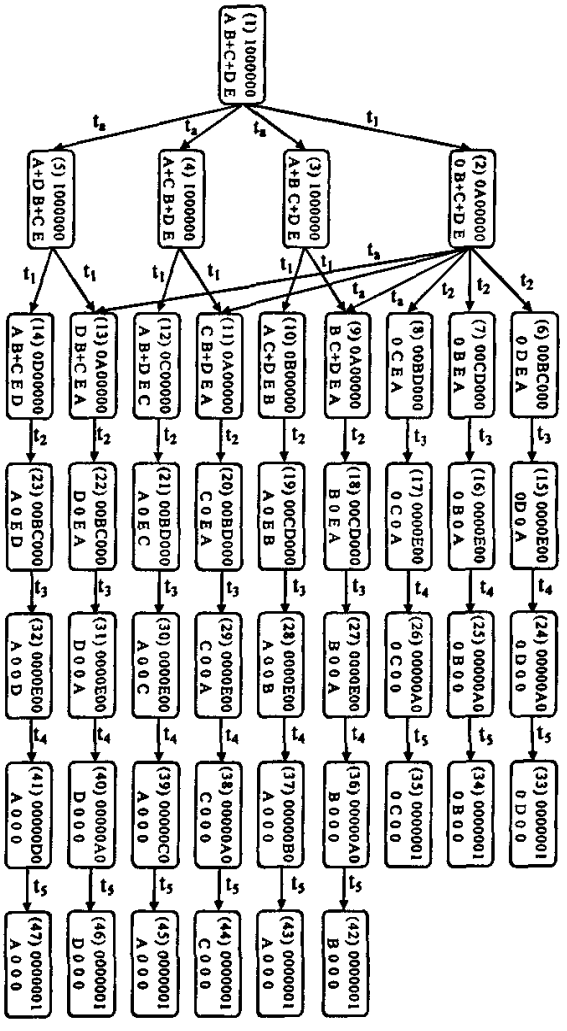


图 5－9　覆盖图

Fig. 5-9 Coverability Graph

(2). (Alice, 论文收集并送审)-(Carl, 论文评审)-(Dick, 论文评审)-(Eric, 评审意见总结)-(Alice, 评审结果反馈);

.
.
.

(6). (Dick, 论文收集并送审)-(Bob, 论文评审)-(Carl, 论文评审)-(Eric, 评审意见总结)-(Dick, 评审结果反馈)。

覆盖图列出了 Petri 网所有可能的状态，因此它也可以用来分析业务流程的可达授权状态，例如上面最后一条执行链就对应了上一小节中的问题的答案，即 Dick 是否会被授权执行论文评审结果反馈的任务。

5．2 网络访问控制策略的建模与分析

中间层策略泛指图 1－1 所示的策略层次结构中的第二和第三层策略，即系统层和抽象设备或应用层策略。中间层策略已涉及到信息基础设施的结构和组成，规定了系统中各组成部件的行为，但不涉及设备和服务的具体内部实现，如设备的型号、性能、接口类型和协议等细节。在分析和验证中间层策略时，一个非常重要的任务是验证中间层策略是否满足或符合高层策略的要求，是否与高层策略相矛盾等等。中间层策略分析的一个特点是由于涉及到抽象的设备和系统，并且相应的策略比高层策略更加明确和具体，因此有关的分析和验证可以比较容易地使用更加形式化的方法来进行。下面以对网络访问控制策略的分析为例，展示中间层策略分析的一个侧面。

防火墙是目前被广泛采用的一种网络保护机制，可以控制对被保护网络的访问和网络内部对外部的访问，所有的访问连接都必须经过防火墙的检查和评估以决定其合法性，从而实施了网络访问控制策略[Anderson1997]。为完成上述任务，防火墙和路由器等相关的网关必须在网络访问控制策略的指导下，进行正确的配置和管理。由于网络和应用的复杂性，对于防火墙和网关配置的分析不能孤立地分析每个防火墙或网关，而是应结合网络的拓扑分析所有防火墙和网关的协作是否满足网络访问控制策略要求。

本节仍采用彩色 Petri 网作为主要的分析工具，通过一个经过扩展的彩色 Petri 网框架，可在系统和抽象设备层上对网络、网络访问控制策略、网关的路由或过滤规则进行建模，利用 Petri 网的覆盖图分析网关的配置是否正确。

由于对彩色 Petri 网框架进行了扩展，下面首先给出相关的定义，然后介绍如何利用该框架对网络及其访问控制策略进行建模和分析。

5．2．1 基本的彩色 Petri 网

由于定义扩展的 Petri 框架的需要，这里重新给出 Petri 网和彩色 Petri 较正式的定义。

定义 5.1 (Petri 网) 一个 Petri 网 PN 是一个四元组，$PN=(P, T, F, M_0)$ 其中：

- P 是一个位置结点(Place)的有限集，$P=\{p_1, p_2, \ldots, p_n\}$；
- T 是一个转换结点(Transition)的有限集，且 $P\cap T=\Phi$, $T=\{t_1, t_2, \ldots, t_m\}$；
- F 是弧(arc)的集合，$F \subseteq (P \times T) \cup (T \times P)$；
- M_0 是一个包(Bag)，代表该 Petri 网的初始标志（Marking）状态。

一个转换结点 t 是一个位置结点 p 的输入转换，当且仅当存在一条从 t 到 p 的有向弧。位置结点 p_i 的输入转换集合表示为 $\bullet p_i=\{t_j\in T \mid (t_j, p_i) \in F\}$。一个转换结点 t 是一个位置结点 p 的输出转换，当且仅当存在一条从 p 到 t 的有向弧。位置结点 p_i 的输出转换集合表示为 $p_i\bullet=\{ t_k\in T \mid (p_i, t_k) \in F\}$。同样 $\bullet t_i$ 和 $t_i\bullet$ 分别表示转换结点 t_i 的输入和输出位置集，即 $\bullet t_i=\{p_j\in P \mid (p_j, t_i) \in F\}$，和 $t_i\bullet=\{ p_k\in T \mid (t_i, p_k) \in F\}$。

任何时候，一个位置结点包含零个或多个令牌(Token)，标志是一个将位置结点映射到令牌数量的函数，$M: P\rightarrow IN$, $IN=\{1, 2, 3, \ldots\}$， Petri 网的标志状态，是所有位置结点标志状态的集合。令牌的数目是变化着的，转换结点是 Petri 网中的活跃因子，它根据激发规则来改变 Petri 网的状态.

定义 5.2 (激发规则 Firing rule) 一个转换要么是可激发的(Enabled)，要么是不可激发的(Disabled):

- 一个转换 t_i 是可激发的，当且仅当 t_i 的每个输入位置结点 p_j 都至少包含一个令牌，即 $\forall p_j\in \bullet t_i$, $M(p_j)>0$，一个可激发的转换可以被激发；
- 如果一个可激发的转换 t_i 被激发，那么 t_i 从每个输入位置结点 p_j 中消耗掉一个令牌，同时向每一个输出位置结点 p_k 中放入一个令牌，即 $\forall p_j\in \bullet t_i$，$\forall p_k\in t_i\bullet$, $M'(p_j)=M(p_j)-1$ and $M'(p_k)=M(p_k)+1$．$M'(p)$ 是位置结点新的标志。

定义 5.3 (可达性 Reachability) 从一个状态 M_1 出，状态 M_n 是可达的当且仅当存在一个或多个激发序列使得状态从 M_1 转变到 M_n，表示为 $M_1 \xrightarrow{\sigma} M_n$, $\sigma=t_1 t_2 t_3 \ldots t_{n-1}$。

定义 5.4 (彩色 Petri 网) 彩色 Petri 网 CPN 是一个五元组，CPN=(PN，Tokens，∑，CR，C)，其中：

- PN=(P, T, F, M_0) 是一个基本 Petri 网；
- Tokens 是令牌的集合，每个令牌都有一个颜色；
- $\Sigma=\{\sigma_1, \sigma_2, \ldots\sigma_n\}$ 是 CPN 所有颜色(类型)的集合；
- CR 是一个将位置结点映射到一个颜色集合的函数，即 $CR(P) \subseteq \Sigma$ and $CR(m(P)) \subseteq CR(P)$，其中 m(P)指位置结点 P 的状态；
- C 是一个将令牌映射到颜色的函数，即 $C(Tokens) \in \Sigma$.

在 CPN 中，每一个令牌都包含一个类型值称为“颜色”，每个 CPN 都定义了一个包含所有可能颜色的集合∑，而每一个位置结点也都定义了一个颜色集 CR(P)，包含所有合法的颜色，CR(m(P))则返回 P 中当前状态下所包含的令牌的颜色。转换结点在激发时根据所消耗的令牌的“颜色”来决定所产生的令牌的“颜色”[Jensen1992]。

5．2．2 网络和网络访问控制策略

目前企业网所采用的防火墙有几种主要类型，如分组过滤器、应用层网关、多层状态检查防火墙等等，本文所讨论的防火墙属于分组过滤防火墙，这也是最常见的防火墙，它实质上是在路由器中加入一个分组过滤器，这种防火墙主要工作在 TCP/IP 协议栈中的 IP 和 TCP 两层，这也是只在 TCP 和 IP 两层上对网络进行建模的原因，网络或子网的底层具体所采用的通信机制，不管是以太网、令牌环网还是交换网等，均被忽略，因此，在本节的模型中，每个子网或网络均可以被看作是一个相对独立的实体。

定义 5.5（网络、子网和网关） 一个企业的网络 $\mathcal{N}=(\mathcal{SN}, \mathcal{G})$，其中：

- $\mathcal{SN}$ 是一个子网的集合，每个子网 $sn_i \in \mathcal{SN}$ 都分配了一个唯一的 IP 地址空间 $IpAdd_{sn} \subset IpAdd$. $IpAdd$ 代表整个 IP 地址空间．有一个特殊的子网“Internet”，代表了该企业网络 $\mathcal{N}$ 以外的所有网络．子网之间通过网关进行连接．
- $\mathcal{G}$ 是一个网关的集合．一个网关 $g_i \in \mathcal{G}$ 可能是一个路由器，也可能是一个包含了分组过滤器的路由器，即防火墙．每个网关至少有两个网络接口，接口的具体定义参见定义 5.7.

定义 5.6 (服务) $\mathcal{SVC}$ 是服务(Service)的集合，$\mathcal{SVC}=\{(pt, pn) | pt \in \mathcal{PT} \wedge pn \in \mathcal{PN}\}$，其中：

- $\mathcal{PT}$ 是协议的集合，$\mathcal{PT}=\{tcp, udp \ldots\}$；

- $\mathcal{PN}$是端口的集合，0≤pn≤65535。

例如，http 服务定义为：http=(tcp, 80)。有的服务可能会使用一个连续的端口地址序列，例如有的音频服务可能需要使用连续 2000 个端口地址，如 4000－6000，这里仅取其中一个作为代表，如最低的端口地址（如 2000），这样做可以简化分析，而并不影响分析的结果，因为在一个企业网的内部，不同的服务之间端口一般不会重叠，而同一个服务，除了端口地址可能不同，但其它性质是相同的。另外必须要指出的是：一个服务代表了一个双向的信息流：从源地址到目标地址（即服务的提供者）的服务请求，和从服务提供者到源地址的服务应答。

路由器根据路由信息将分组从一个网络（子网）传递到另一个网络（子网），路由信息以（dest, nxhp）序偶的形式保存在路由表中，其中 dest 是目标网络的 IP 地址，nxhp 是到 dest 的路径上的“下一个”路由器的 IP 地址，称之为“下一跳（站）”，路由器通过与 nxhp 有直接连接的接口将分组送出。

防火墙根据其规则库中的过滤规则来过滤分组。一个过滤规则包括四个基本的元素：（srce，dest，serv，action），其中 srce 和 dest 分别代表源和目的 IP 地址（一般是 IP 地址的集合），serv 代表相应的服务，action 为“permit”（允许）或“deny”（禁止），表示允许或禁止相应的信息流通过。更常见的过滤规则还包括接口和方向信息，表达了分组通过哪个接口进入（in）或离开（out）防火墙，实质上它们提供的是路由信息。

定义 5.7（接口、规则集和规则）接口是网关的一个组成部分，每个接口分配了一个唯一的 IP 地址，该地址隶属于它所直接相连的网络的 IP 地址空间，每个接口都有一个规则集 rs⊆ $2^{\mathcal{R}}$，其中 $\mathcal{R}$是规则的总和，每个接口的规则集是相应的网关的规则库的一个子集。每个规则 r ∈ $\mathcal{R}$包含了 5 个元素：（srce，dest，serv，action，dirc），其中：

- srce 是源地址（集）；
- dest 是目标地址（集）；
- serv 是相应的服务；
- action 是 “*permit*” 或者 “*deny*”；
- dirc 是 “in” 或者 “out”，“in” 表示分组通过该接口进入网关，“out” 表示分组通过该接口离开网关。

对路由器来说，接口的规则集中的规则通常为(*, dest, *, *permit*, *in/out*)的形式，“*”是通配符，表示所有可能的选择。一个路由器只关心如何向前传递一个

分组，因此在路由器中，一个目标地址有一个出口接口，而其余的接口则是入口接口。一个接口的规则集可以从路由器的路由表中得出。例如，在图 5－10 中所示的网络中，路由器 R1 的路由表中有一项（D，I_{21}），因此 R1 知道目标地址为 D 的分组的下一站是接口 I_{21} 所在的路由器 R2，由此可以得出 3 个规则：接口 I_{11} 和 I_{13} 规则集中的 2 个相同的规则(*, D, *, *permit, in*)，和接口 I_{12} 的规则集中的 1 个规则(*, D, *, *permit, out*)。

对于防火墙来说，接口的规则集中的规则应该是(srce, dest，serv，*permit/deny, in/out*)的形式。可以从防火墙的规则库中通过收集与该接口有关的规则而直接获得。通常，对于绝大多数类型的防火墙来说，其规则库中的规则是对顺序敏感的，即遵循“首先匹配的规则将适用”的原则。

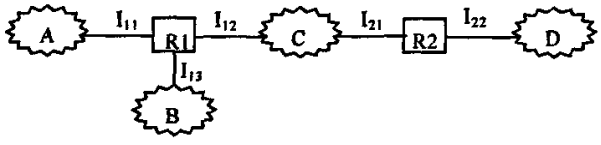


图 5－10　接口和规则

Fig. 5-10 Interface and Rule

对于图 2－6 所示的网络实例，表 5－1 列出了各子网所提供的服务和相应的服务器。

表 5－1. 实例网络中的服务器和提供的服务

Tab. 5-1 The Servers and Services in the Example Network

子网	服务器	服务	(pt, pn)
DMZ	8 0.1 4 8 0.1.5	http ftp	(*tcp*, 80) (*tcp*, 21)
R(Research)	8 0 4 12	ftp	(*tcp*, 21)
A(Accounting)	8 0 2 3	dbapp	(*tcp*, 265)

该机构与网络访问控制有关的高层策略如下：

- 所有的主机，包括 Internet 中的主机都可以访问 DMZ 中所提供的 http 和 ftp 服务；

```
auth+ DMZServHttp {
```

```
    subject        /*/Host/*;
    target         /DMZ/Services/8.0.1.4/Service/http;
    action         access();
}

auth+ DMZServFtp {
    subject        /*/Hosts/*;
    target         /DMZ/Servers/ 8.0.1.5 /Service/ftp;
    action         access();
}
```

- IP 地址为 8.0.4.12 的服务器提供的 ftp 服务对 DMZ 以外所有的主机都是可用的；

```
Auth- ResearchServFtp1 {
    subject        /DMZ/Hosts/*;
    target         /Research/ Servers/ 8 0.4 12/Service/ftp;
    action         access();
}

auth+ ResearchServFtp2 {
    subject        /*/Hosts/*;
    target         /Research/Servers/ 8 0 4.12/Service/ftp;
    action         access();
}
```

- IP 地址为 8.0.2.3 的服务器提供的 dbapp 服务只对 A 中的所有主机和 M 中的地址为 8.0.3.11 的主机是可用的。

```
auth+ AccountingServDbapp {
    subject        /Acounting/Hosts/*+/Management/Hosts/8.0.3.11;
    target         /Acounting/Servers/8.0.2.3/Service/dbapp;
    action         access();
}
```

5．2．3 一个扩展的彩色 Petri 网框架

在传统的网络建模方法中，网络拓扑一般被表示为一个无向图，图中的顶点代表一个或一组主机，而顶点之间的弧代表主机或网络之间的通讯线路[Permpoontanalarp2001]，这是因为一个通讯线路一般允许两个方向的数据流。而 Petri 网是一种有向图，如果被用来对网络建模，必须采用某种方法将弧的方向确定下来，

这个问题将在后面再讨论。这里首先讨论如何将网络及访问控制策略映射到 Petri 网。当然，在弧的方向未确定下来之前，所得到的模型是不完整的。

定义 5.8（映射规则） 一个网络 $\mathcal{N}=(\mathcal{SN}, \mathcal{G})$ 根据下列映射规则映射到一个彩色 Petri 网 CPN=(PN, Tokens, Σ, CR, C):

- 每个子网 $sn \in \mathcal{SN}$ 映射到一个位置结点 p_S，该位置结点定义了一个颜色集 CR(p_S)，包含了所有合法的颜色，每个合法的颜色代表了一个与 sn 有关的合法的信息流，该信息流要么起源于 sn，要么终止于 sn。p_S有一个属性 p_S ->$IpAdd_{sn}$对应于分配给该子网的 IP 地址空间。
- 每个网关被映射到一个位置结点 p_g和 n 个转换结点 t_{gi}，i = 1，2，...，n。n 是网关中的接口的数量。每个位置结点 p_g都定义了一个颜色集，其含义和 p_S相同。每个转换结点 t_{gi}都定义了一个规则集，对应于相应的接口的规则集。
- 每个信息流被映射到一种颜色。颜色定义为：δ=(srce, dest, serv)，其中 srce 和 dest 分别是源和目的 IP 地址，serv 是相关的服务。
- 一个分组被映射到一个令牌，每个令牌定义了一种颜色，代表了该分组所属的信息流。
- 子网和网关之间的通讯线路映射成它们之间的一个弧，在网关内部的通信通道也被映射成每个 t_{gi}和 p_g之间的弧。

从上面的定义可以看出，CPN 模型中有两种类型的位置结点：一种是 $\mathcal{P}_S$，代表子网位置结点 p_s；一种是 $\mathcal{P}_G$，代表网关位置结点 p_g。

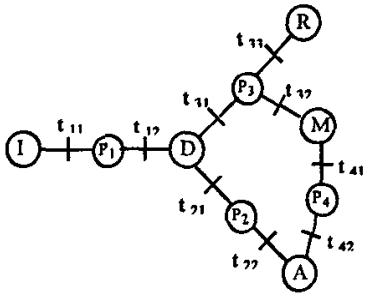


图 5－11 示例网络的 Petri 网模型（弧的方向尚未确定）

Fig. 5-11 The Petri Net Model of the Example Network with Arcs Un-derminted

图 5－11 给出了示例网络的 CPN 模型，注意图 5－11 并不是一个完整的 CPN，因为弧的方向尚未确定。有关的访问控制策略被翻译成 5 个子网位置结点的颜色集中的颜色，如表 5－2 所示。每个网关位置结点 p_g 同样也应该有一个颜

色集，其中的颜色对应于网关所允许的提供给其它网关的服务，或向系统管理员提供的进行远程控制的接口。为简明起见，它们未在表中给出。

表 5－2 位置结点的颜色集

Tab. 5-2 The Color Sets of Places

子网	A (Accounting)	M (Management)	R (Research)	D (DMZ)	I (Internet)
颜色集	c1: (8 0.3.11, 8.0.2 3, dbapp) c2: (A, I, http) c3. (A, 8.0.1 4, http) c4 : (A, 8.0.1.5, ftp) c5 (A, 8 0 4.12 ftp)	c1· (8.0 3.11, 8 0 2 3, dbapp) c2 (M, I, http) c3 (M, 8.0.1.4, http) c4. (M, 8 0.1.5, ftp) c5. (M, 8.0.4.12 ftp)	c1. (M, 8 0.4 12, ftp) c2 (A, 8 0.4.12 ftp) c3 (I, 8 0 4 12 ftp) c4· (R, 8 0 1 4, http) c5. (R, 8.0.1 5, ftp) c6· (R, I, http)	c1:(*, 8 0.1 4, http) c2: (*, 8.0.1.5, ftp)	c1: (*, I, http) c2 (I, 8 0 4 12 ftp) c3. (I, 8 0 1 4, http) c4: (I, 8.0.1.5, ftp)

定义 5.9（src，des，svc，act 和 dir）函数 src、des 和 svc 分别返回某个颜色或规则的 srce、dest 和 serv 部分，函数 act 和 dir 分别返回某个规则的 action 和 dirc 部分。即：

src: $\sum \cup \mathfrak{R} \rightarrow 2^{IPAdd}$

des: $\sum \cup \mathfrak{R} \rightarrow 2^{IPAdd}$

srv: $\sum \cup \mathfrak{R} \rightarrow SVC$

act: $\mathfrak{R} \rightarrow \{permit, deny\}$

dir: $\mathfrak{R} \rightarrow \{in, out\}$

大多数防火墙采纳了封闭策略(Closed Policy)，即除非明确定义了一条规则允许某个信息流通过，信息流在缺省规定下是被禁止的。为表达方便，在下文中，缺省的“deny”规则没有明确列出。

表 5-3 列出了所有本文的网络示例中有关的规则集和其中的规则。

表 5－3 实例网络模型中的各个转换结点的规则集和规则

Tab. 5-3 Rule Base and Rules of the Transitions in the Example Network

规则集	过　滤　规　则
rs_{11}	r1: (*, 8.0.1.4, http, *permit*, *in*)　　r2: (*, 8.0.1.5, ftp, *permit*, *in*) r3: (D, I, http, *deny*, *out*)　　r4: (8.0.1.0-8.0.4.255, I, http, *permit*, *out*)

rs_{12}	r1. (I, 8.0.1.4, http, *permit, out)*　　r2 (I, 8.0 1.5, ftp, *permit, out*) r3: (D, I, http, *deny, in*)　　r4 (8 0 1 0-8.0.4 255, I, http, *permit, in*)
rs_{21}	r1: (M, 8.0.2.3, dbapp, *permit, in*)　　r2: (A, I, http, *permit, out*) r3: (A, 8.0.1.4, http, *permit, out*)　　r4: (A, 8.0.1.5, ftp, *permit, out*) r5: (A, 8 0.4.12, ftp, *permit, out*)
rs_{22}	r1: (M, 8.0.2.3, dbapp, *permit, out*)　　r2: (A, I, http, *permit, in*) r3: (A, 8.0.1.4, http, *permit, in*)　　r4: (A, 8.0.1.5, ftp, *permit, in*) r5: (A, 8.0.4.12, ftp, *permit, in*)
rs_{31}	r1: (D, 8.0.4.12, ftp, *deny, in*)　　r2: (8.0.1.0-8.0.4.255, 8.0.4.12, ftp, *permit, in*) r3: (R, 8.0.1.4, http, *permit, out*)　　r4: (R, 8.0.1.5, ftp, *permit, out*) r5: (*, I, http, *permit, out*)　　r6: (M, 8.0.1.4, http, *permit, out*) r7: (M, 8.0.1.5, ftp, *permit, out*)　　r8: (M, 8.0.2.3, dbapp, *permit, out*) r9: (I, 8.0.4.12, ftp, *permit, in*)
rs_{32}	r1: (8.0.1.0-8.0.4.255, 8.0.4.12, ftp, *permit, in*)　　r2: (M, 8.0.1.4, http, *permit, in*) r3: (M, 8.0.1.5, ftp, *permit, in*)　　r4: (M, 8.0.2.3, dbapp, *permit, in*) r5: (*, I, http, *permit, in*)
rs_{33}	r1: (D, 8.0.4.12, ftp, *deny, out*)　　r2: (8.0.1.0-8.0.4.255, 8.0.4.12, ftp, *permit, out*) r3: (R, 8.0.1.4, http, *permit, in*)　　r4: (R, 8.0.1.5, ftp, *permit, in*) r5: (*, I, http, *permit*, in)　　r6: (I, 8.0.4.12, ftp, *permit, out*)
rs_{41}	r1: (8.0.3.11, 8.0.2.3, dbapp, *permit, in*)
rs_{42}	r1: (8.0.3.11, 8.0.2.3, dbapp, *permit, out*)

下面给出 CPN 网络模型的几个性质：

性质 5.1 对于转换结点 t_{gi}（对应于接口 i）的规则集 rs_i 中的一个规则 r_k，至少有一个对等（peer）规则 r_l，r_l是同一网关另外一个接口 j 对应的转换结点 t_{gj} 的规则集 rs_j 中的一个规则，它们之间满足下面的关系：

$\forall r_k \in rs_i, \exists r_m \in rs_j$: $src(r_k) = src(r_m)$ and $des(r_k) = des(r_m)$ and $srv(r_k) = srv(r_m)$ and $act(r_k) = act(r_m)$ and $dir(r_k) \neq dir(r_m)$.

性质 5.2 在一个规则集 rs_k 中，没有两个规则在方向上是相互矛盾的。即：

$\forall r_i \in rs_k, \neg\exists r_j \in rs_k$: $des(r_i) \cap des(r_j) \neq \Phi$ and $dir(r_i) \neq dir(r_j)$

这是由网关自身的要求所决定的，如果矛盾确实存在，则以排在最前面的规则为准。

性质 5.3 在一个网络的 CPN 模型中，每个转换有且仅有一个输入和一个输出位置结点，其中一个位置结点属于子网类型位置结点，另外一个属于网关类型位置结点，至于哪一个属于输入，哪一个属于输出，要等弧的方向确定下来才能知道。相反，一个位置结点可以有多个输入和输出转换。

绝大多数的企业的网络所采用的内部网关协议 IGP（Interior Gateway Protocol）如 RIP(Routing Information Protocol)和 OSPF(Open Shortest Path First)都能保证内部网络的路由不存在回路[Stevens1994]，因此可以得出网络的下面的性质。

性质 5.4 所有需要经过两个相同的网关，如 A 和 B，并有着相同目标地址的分组在某段时刻移动的方向相同，要么从 A 到 B，要么从 B 到 A，否则路由将出现回路，这是 IGP 所不允许的。

CPN 中弧方向的确定是模型中最关键和有趣的部分。弧的方向是根据所要进行的分析的类型而动态地确定的。这里鉴别出两种类型的分析：

1. 某个子网中的一个或一组主机能访问什么服务？
2. 谁能访问某个子网（或确切地说，该子网中的某个主机）所提供的某个服务？

在第一种分析中，弧方向的确定算法如下：

算法 5.1(a). 当分析子网 A 中的主机所能够访问的服务时，CPN 弧方向的确定算法：

foreach gateway (p_g, (t_1, t_2, ...t_n))

/** 根据定义 4.1，一个网关被映射为一个 p_g 和 n 个 transitions t_1, t_2, ...t_n。 */

foreach t_i of the gateway

{

if $\exists r_j \in rs_i$ and act (r_j)=*permit* and dir (r_j)=in and (src (r_j) $\cap$ A $\neq \Phi$ and src (r_j) $\neq$ *)

/** rs_i 是 t_i.的规则集，（src (r_j) $\cap$ A $\neq\Phi$ and src (r_j) $\neq$*) 表示 r_j 是一个过滤规则。一个过滤规则暗示一个双向的信息流 */

then $p_g \in t_i^{\bullet}$ and $p_{si} \in {}^{\bullet}t_i$

/** p_{si} 指接口 i 所直接相连的子网所对应的位置结点，根据性质 5.3，可知只有一个这样的位置结点存在。 */

elseif $\exists r_k \in rs_i$ and act (r_k)=*permit* and dir (r_k) = *out* and (src (r_k) $\cap$ A $\neq \Phi$ and src (r_k) $\neq$ *)

then $p_g \in {}^{\bullet}t_i$ and $p_{si} \in t_i^{\bullet}$

elseif ∃ r_l ∈rs_i and act (r_l)=*permit* and dir (r_l)= *out* and des (r_l) ∩ A ≠ Φ
then p_g ∈ t_i• and p_{si} ∈ •t_i
elseif ∃ r_m ∈rs_i and act (r_m)=*permit* and dir (r_m)=in and des (r_m) ∩ A ≠ Φ
then p_g ∈ •t_i and p_{si} ∈ t_i•
else delete the arcs between (t_i, p_g) and (t_i, p_{si})

}

该算法分别遍历每个转换节点的规则集，检查是否存在一个“permit”规则，其源地址部分与 A 向匹配，即 src (r_j) ∩ A ≠ Φ，如是，则表明该接口位于从 A 中发出的分组所经过的路径上，因而弧的方向可根据该规则的方向部分来确定。性质 5.2 可保证弧方向是可确定的。如果找不到这样的规则，算法将继续检查是否存在一个“permit”规则，其目标地址部分与 A 相匹配。这是因为一个通讯通道通常是允许两个方向的数据流的，如果一个网关被设置成允许向前投递目的地为 A 的分组，它很有可能也被允许投递源于 A 的分组，至少它可以被配置成这样。本节所采用的分析的方法决定这样假设不会影响对网关配置的分析。最后，如果接口和 A 没有任何关系，则可将有关的弧删除。

在第二种分析中，确定弧的方向的算法和第一种实质上是相同的，仅仅是检查的顺序不同，这是因为后面所采用的分析方法决定的。其算法如下：

算法 5.1(b). 分析谁能访问子网 A 所提供的某个服务时，CPN 中弧的方向的确定算法：

/** 本算法所确定的弧的方向与相应的分组实际传递方向相反，这与后面所采用的分析方法相对应。*/
foreach gateway (p_g, (t_1, t_2, ...t_n))
foreach t_i of the gateway
{
if ∃ r_j ∈rs_i and act (r_j)=*permit* and dir (r_j)= *out* and dst (r_j) ∩ A ≠ Φ
/** *rs_i is the rule-set associated with t_i* */
then p_g ∈ t_i• and p_{si} ∈ •t_i
/** p_{si} 是与接口 i 直接相连的子网所对应的位置结点，根据性质 4.3，每个 i 只有一个 p_{si}。*/
elseif ∃ r_k ∈rs_i and act (r_k)=*permit* and dir (r_k)=*in* and dst (r_k) ∩ A ≠ Φ
then p_g ∈ •t_i and p_{si} ∈ t_i•
elseif ∃ r_l ∈rs_i and act (r_l)=*permit* and dir (r_k)=in and (src (r_l) ∩ A ≠ Φ and src (r_l) ≠ *)
/** rs_i 是 t_i.的规则集，（src (r_j) ∩ A ≠Φ and src (r_j) ≠*) 表示 r_j 是一个过滤规则。一个过滤规则暗示一个双向的信息流 */
then p_g ∈ t_i• and p_{si} ∈ •t_i
elseif ∃ r_m ∈ rs_i and act (r_m)=*permit* and dir (r_m) = *out* and (src (r_m) ∩ A ≠ Φ and src (r_m) ≠ *)
then p_g ∈ •t_i and p_{si} ∈ t_i•
else delete the arcs between (t_i, p_g) and (t_i, p_{si})

```
}
/**根据性质 5.2，没有两个规则是相互矛盾的，可保证弧方向是可确定的。*/
```

上述两个算法的复杂度为 O (nml)，其中 m 为网络中网关的数量，n 为网关中接口的数量，l 是规则集中规则的数量，通常 m 和 n 较小。性质 5.4 保证了根据上述俩个算法确定弧方向后 CPN 模型中没有环路存在。

每个颜色的 srce 和 dest 部分分别对应两组 IP 地址，因此一般一个颜色可以看作是由一组更小的颜色组成，称最小的颜色单位为原子颜色，其定义如下：

定义 5.10（原子颜色）一个颜色δ的原子颜色δ^{A}定义为：

$$\delta^{A}(\delta)=\{(ip_s, ip_d, s)|\ ip_s \in src(\delta) \wedge ip_d \in des(\delta) \wedge s=svc(\delta)\}$$

即每个原子颜色的 srce 和 dest 部分只包含单个 IP 地址。

由于颜色的这种特性，一个颜色是否属于某个颜色集不能简单地根据是否能在该颜色集中找到一个匹配来确定。因此，有下面的定义。

定义 5.11（⌉）一个颜色δ属于某个颜色集 cs 定义为：

$\rceil : \Sigma \times 2^{\Sigma} \rightarrow \{true, false\}$

$\delta \in \Sigma$, $cs \in 2^{\Sigma}$, $\delta \rceil cs$ 为真，当且仅当

$\forall \delta_i^s \in \delta^A(\delta), \exists \delta_j \in cs: \delta_i^s \in \delta^A(\delta_j)$　　(δ_i^s为原子颜色)

即δ中的每个原子颜色δ_i^s，都能在 cs 中找到一个颜色δ_j，其原子颜色中包含δ_i^s。

每个令牌都定义了一个颜色代表了它所属的信息流。下面给出与一些与令牌和规则相关的操作的定义，在定义激发规则时，需要用到这些操作：

定义 5.12（(）操作（ 定义如下：

(: Tokens X ℜ→Tokens

for tok, tok' ∈ Tokens, r ∈ ℜ, srv(C(tok)= srv(r):

tok' = tok (r

其中：

src(C(tok'))=src(C(tok)) ∩ src(r)　and

dst(C(tok'))=dst(C(tok)) ∩ dst(r)　and

srv(C(tok')= srv(C(tok)) ∩ srv(r)

if src(C(tok'))= Φ or dst(C(tok'))= Φ

then tok' = nil

(nil 表示没有令牌产生。C(tok)返回 tok 的颜色。tok (r 将产生一个新的令牌，其颜色为原来的令牌和规则中颜色部分的交集。)

定义 5.13 (⊖) 操作 ⊖ 定义如下:

⊖ : Tokens X $\mathfrak{R}$→Tokens

for tok, tok' ∈ Tokens, r ∈ $\mathfrak{R}$, srv(C(tok)= srv(r):

tok' = tok ⊖ r

其中:

src(C(tok'))=src(C(tok)) - src(r)　　and

dst(C(tok'))=dst(C(tok)) - dst(r)　　and

srv(C(tok')= srv(C(tok)) ∩ srv(r)

if src(C(tok'))= Φ or dst(C(tok'))= Φ

then　tok'=nil

(操作 tok ⊖ r 将产生新的令牌，其颜色为原来令牌的颜色中去掉与规则中颜色部分相重叠的部分。)

定义 5.14 (⊕) 操作 ⊕ 定义如下:

⊕ : Tokens X $\mathfrak{R}$→ Tokens

for tok ∈ Tokens, r ∈ $\mathfrak{R}$, srv(C(tok)= srv(r) and act(r)=*permit* ,

tok'=tok ⊕ r =tok ⊖ r_a ⊖ r_b ⋯ ⊖ r_i (r　where

r_a, r_b, ⋯ r_i 是规则集中所有位于 r 之前且 srv(r_i) =srv(C(tok) and act(r_i)=*deny* 的规则。

如果 tok' ≠ nil, 则称令牌 tok 与 "permit" 规则 r 相匹配。

(令牌与 "permit" 规则相匹配表示该规则可应用于相应的信息流)

至此，可以给出扩展的 CPN 框架中剩下的定义。

定义 5.15 (可激发的转换) 一个转换 t_i 在一个标志状态 M 下是可激发的，当且仅当:

对于 t_i 的输入位置结点 p_i∈ $^\bullet t_i$, ∃ tok_j∈ M(p_i), ∃ r_k ∈ rs_i, srv(C(tok_j)= srv(r_k) and act(r_k)= *permit* :

tok_j ⊕ r_k ≠ nil and ((p_i ∈ $\mathfrak{P}_G$ and dir(r_k)= *out*) or (p_i ∈ $\mathfrak{P}_S$ and dir(r_k)=*in*))

(注意 t_i 只有一个输入位置结点 p_i。)

为方便起见，称 tok_j 是转换 t_i 的使能令牌。

定义 5.16 (激发规则) 一个可激发的转换可以被激发，激发规则如下:

$$M'(p)=\begin{cases} M(p)-\{tok_j\} & \text{if } p \in {}^\bullet t_i \text{ and } tok_j \text{ 是 } t_i \text{ 的使能令牌} \\ M(p)+\{tok_1', tok_2', \ldots, tok_m'\} & \text{if } p_i \in t_i^\bullet \\ M(p) & \text{其它} \end{cases}$$

其中：M'(p)是 p 在 t_i 激发后的标识，m 是规则集 rs_i 中与 tok_j 相匹配的“permit”规则的数目，$tok_k'=tok_j \oplus r_k$。

值得注意的是规则集 rs_i 中与 tok_j 相匹配的“permit”规则间可能存在重叠，因此在转换结点激发时可能会产生颜色相互重叠的令牌，即它们的颜色包含有相同的原子颜色，这种情况是可接受的因为不会影响 CPN 模型可达性分析的结果。

5．2．4 网关配置分析

下面介绍如何使用 CPN 的覆盖图（Coverability Graph）来分析网关（包括路由器和防火墙）的配置是否实现了组织的网络访问控制策略，并且能够提供问题的解决方案。网关配置分析主要有两个目的：

1．检查是否有非法的访问被允许

如果在当前的网关配置下，非法的访问被允许，则过滤规则必须设定的严格一些。问题是哪一个防火墙是阻塞非法访问的最佳位置，对网关配置的分析应该能够解答这个问题。

2．检查是否有合法的访问被禁止

如果合法的访问被禁止，过滤规则必须设定的宽松一些。网关配置分析应该能告诉哪些防火墙的规则需要改动。

对于前面提到过的两种类型的网关配置分析，下面将首先详细讨论第一种类型的分析。

5．2．4．1 分析某个子网中的一个或一组主机能够访问的服务

如果要分析子网 *S* 中的主机能够访问什么服务，可采用下面的步骤（假设网络的 CPN 模型已经建立，只有弧的方向还未确定）：

1．根据算法 5.1(a)确定弧的方向。

2．初始化 CPN，在位置结点 *S* 中放入一个令牌，令牌的颜色为(*S*, *, *)，其它位置结点中没有令牌。

3. 构造 CPN 的覆盖图。覆盖图中每个节点对应于 Petri 网的一个标志状态，节点中包含 m 项，m 是 Petri 网中位置结点的数目，每一项代表了对应的位置结点的标志状态。节点之间的箭头标有引起对应的标识状态变化的转换结点。图中的叶子节点对应于该 Petri 网的一个终端状态，即没有可激发的转换的标志状态。为降低覆盖图的复杂度，这里对其进行了简化，简化去掉了一些中间节点，这些中间节点起源于同一节点（源节点），并终止于同一节点（终止节点）。之所以产生这些节点以及相应的从源节点到终止节点的不同的路径，只是由于相同的可激发的转换的激发顺序不一样而已。这些中间节点和相应的箭头被一个箭头取代，并标以所有相关的转换的和。这种简化不会影响对 CPN 可达性的分析，因为分析人员只对覆盖图中的叶子节点感兴趣。

4. 通过对覆盖图的分析，可以达到前面所提出的网关分析的目的。

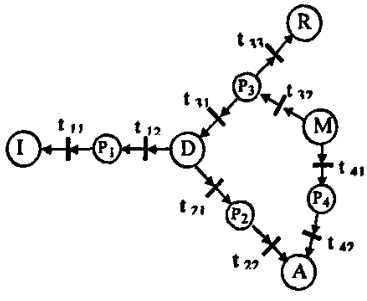


图 5－12 分析 M 中的主机所能访问的服务时完整的 CPN

Fig. 5-12 The Complete CPN when Analyzing What Service Host in M can Access

在进一步展开讨论之前，先给出下面两个定义：

定义 5.17(a)（沉淀和未沉淀的令牌）在标志状态 M 下，一个位置结点 p 中的令牌 tok 如果满足：$des(C(tok)) \subseteq p->IpAdd_{sn}$（$p->IpAdd_{sn}$是位置 p 的一个属性，代表 p 的 IP 地址空间），则称该令牌是沉淀的，否则令牌是未沉淀的。一个沉淀的令牌表示其所在的位置结点对应于相应的分组的目标网络或子网。

定义 5.18（安全和非安全投放的令牌）如果一个位置结点 p 中的令牌 tok 满足：C (tok) ⌉ CR (p)，称 tok 被安全地投放到 p 中；否则称为非安全的投放。

例如，对于图 5－11 所示的 CPN 网络模型，如果要分析子网 M 中的主机所能访问的服务，可根据算法 5.1(a)可将 Petri 网中弧的方向确定下来，如图 5－12 所示。通过将一个颜色为（M, *, *）的令牌投放到位置结点 M 中而初始化该 CPN，

然后构造其覆盖图，如图 5－13 所示。图中每个节点包含两行内容，第一行有 5 项，分别对应 5 个子网位置结点 A、M、R、D 和 I，第二行有 4 项，分别对应 4 个网关位置结点 p_1、p_2、p_3 和 p_4。

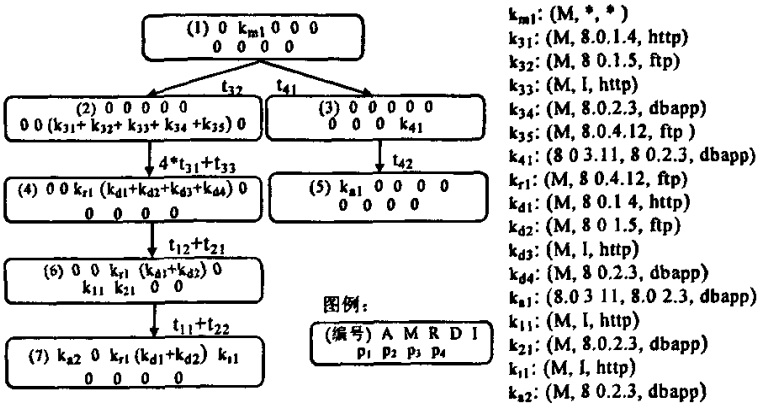


图 5－13 图 5－12 所示的 CPN 的覆盖图

Fig. 5-13 The Coverability Graph of the CPN in Figure 5-12

对图 5－13 所示的覆盖图进行分析，这里讨论以下几种情形：

(1). 在覆盖图中的叶子节点所对应的终端标识状态中，如果某个位置结点中的令牌是安全投放的沉淀令牌，则表示与令牌所代表的信息流有关的过滤或路由规则是正确地被配置在有关的网关中。例如，图 5－13 的覆盖图中有两个叶子节点，即节点 5 和 7。在节点 5 中，令牌 k_{a1}: (8.0.3.11, 8.0.2.3, dbapp) 被安全投放到位置 A 中，此时 k_{a1} 是沉淀的。从根节点到叶子节点的路径代表了对应的分组所走过的路径，因此可以得出结论，在这条路径上的所有网关中，对应于 k_{a1} 的颜色所代表的信息流的过滤规则均已被正确配置。

(2). 在叶子节点中如果一个沉淀的令牌是非安全投放的，则表示存在着非授权的访问被允许的情况，因此相应的过滤规则必须被设定的严格一些。最合适的阻塞点应该是该路径中的第一道防火墙。例如在图 5－13 的节点 7 中，令牌 k_{a2}: (M, 8.0.2.3, dbapp) 被非安全地投放到 A 中，这代表 M 中所有的主机都被允许访问 dbapp 服务，违反了企业的安全访问策略。相关的过滤规则应该被重新设定的严格一些以阻止 M 中除 8. 0. 3. 11 以外的主机的访问。尽管在分组经过的路径上，只需要重新设定一个网关的过滤规则即可，但建议该路径上所有的网关都应该重

新设定相应的过滤规则以确保其安全性。所涉及的网关可以从根节点到叶子节点的路径中所激发的相应的转换结点中得出。

从上面的分析可以看出，从 M 到 A 存在两条路径，都必须被正确地配置，否则就会出现安全漏洞。

(3). 对于叶子节点中未沉淀的的令牌，如果根据安全策略其所代表的信息流是合法的或部分合法的，则说明合法的访问被当前网关的配置所禁止，这时必须放松相应的过滤规则。在叶子节点中，未沉淀的令牌只会在子网位置结点中发现，这是因为根据性质 5.1，如果分组能够进入网关，那么它总可以从另外一个接口离开。需要放松过滤规则的网关应该是到目标的路径上的下一个网关，它是该未沉淀的令牌所在的位置结点的输出转换结点所对应的网关。如果该位置结点有两个以上的输出转换（位置结点允许有多个输出或输入转换），则规则集中有满足：$\exists\ r_j \in rs_i$, $des(r_j) \cap des(C(tok)) \neq \Phi$ and $dir(r_j)=in$ 或者$\exists\ r_k \in rs_k$, $des(r_k) \cap (C(tok)) \neq \Phi$ and $dir(r_k)=out$ 条件的转换都应该应该调整过滤规则的转换。过滤规则调整后，因为又存在着可激发的转换，则覆盖图可以被延伸，这个过程可以被重复直至相应的令牌被沉淀，在下一小节的讨论中，将举例说明这种情况。

这个过程也可以用来获取防火墙的过滤规则，从头开始配置防火墙。

(4). 对于叶子结点中未沉淀的令牌，如果根据安全策略，其所属的信息流是非法的，则表明非法的访问被成功地拦截。

5．2．4．2 分析谁能够访问某个子网所提供的服务

如果要分析谁能够访问某个子网所提供的服务，如 *S* 中主机 *hx* 所提供的 *svcx* 服务，可采用以下步骤：

1．根据算法 5.1(b) 确定弧的方向。

2．初始化 CPN，在位置结点 *S* 中放入一个令牌，令牌的颜色为(*, *hx*, *svcx*)，其它的位置结点中没有令牌。

3．采用与前面相同的方法构建覆盖图。通过对覆盖图的分析来完成验证任务。在这里的分析中，令牌传递的方向和实际分组传递的方向是相反的。然而，前面已经提到一个合法的访问往往意味着双向的信息流，即服务的请求和应答。在某些防火墙的实现中，要求管理员为每个合法的访问配置两条过滤规则：一条应用于向服务提供者发出的请求；一条应用于服务提

供者的应答[Stevens1994]。在本模型中，两条规则合并为一条并代表双向信息流，颜色的含义与此相同。假如服务请求和应答采用不同的路径，分析一个方向的可达性是可行的，这是因为 Petri 网的覆盖图能够覆盖所有可能的路径。

这里对沉淀和未沉淀的令牌的定义与前面稍有不同：

定义 5.17(a).（沉淀和未沉淀的令牌）在标识状态 M 下，一个位置结点 p 中的令牌 tok 如果满足：src (C (tok)) ⊆ p->IpAdd$_{sn}$（p->IpAdd$_{sn}$是位置 p 的一个属性，代表 p 的 IP 地址空间），则称该令牌是沉淀的，否则称令牌是未沉淀的。一个沉淀的令牌表示其所在的位置结点对应于相应分组的源网络或子网。

例如，如果要分析谁能够访问子网 R 中的主机 8.0.4.12 提供的 *ftp* 服务，首先根据算法 5.1(b) 确定弧的方向，如图 5－14 所示。图中的结点 p_4, t_{41}, t_{42} 和相应的弧用虚线表示，因为根据网络配置，这是 M 和 A 之间的专用连接，任何到 R 的连接都不允许经过该连接。接下来初始化 CPN 模型，将一个令牌(*, 8.0.2.4, *ftp*)放入到位置 R 中，构建覆盖图如图 5－15 所示。

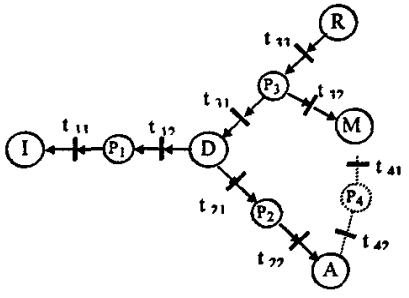


图 5－14 分析谁能够访问 R 中的服务时的完整的 CPN

Fig. 5-14 The Complete CPN when Analyzing Who Can Access the Services Provided in R

在图 5－15 中，唯一的叶子节点 5 中，位置 D 中有一个未沉淀的令牌 k_{d2}: (I, 8.0.4.12, *ftp*)，根据企业的网络访问控制策略知道，相应的网络访问是合法的。由于相应的网关配置禁止该访问，因此需要放松有关的规则。通过检查位置节点 D 的所有输出转换中的规则集，发现 t_{12} 的规则集 rs_{12} 中有一条规则 r_3 的目的地址部分为“I”而且方向部分为“in”，所以应该在 rs_{12} 中增加一条规则(I, 8.0.4.12, *ftp*, *permit*, *out*)，相应地根据性质 5.1，rs_{12} 中也应加入一条对等规则(I, 8.0.4.12, *ftp*, *permit*, *in*)。这时转换结点 t_{12} 变成可激发的，从而可将覆盖图延伸下去至新的叶

子节点 7。在节点 7 中，令牌 k_{11} 是安全沉淀的，表示相应的过滤规则配置正确。这个过程经过迭代，也可以用来从头生成过滤规则。

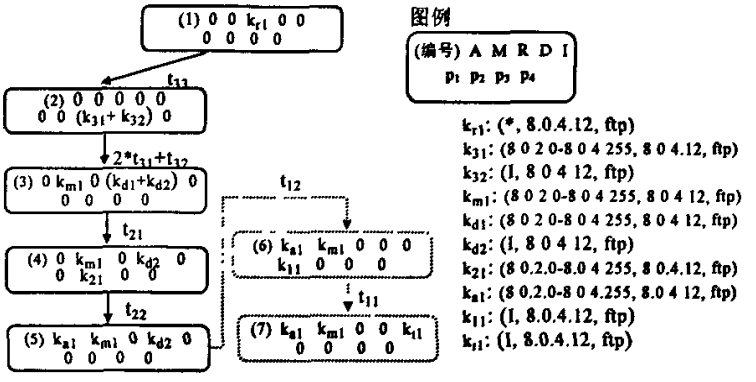


图 5－15 图 5－14 所示的 CPN 模型的覆盖图

Fig.5-15 The Coverability Graph of the CPN in Figure 5-14

5．3 防火墙过滤规则的建模、分析和优化

低层策略泛指设备或应用实例层策略。这些策略面向具体的系统设备和应用实例，可由设备直接理解和实施，与设备或应用的实际配置策略或配置文件相对应。例如下面要分析的防火墙过滤规则与现实中防火墙配置中的过滤规则是一一对应的，可以直接用来配置防火墙而无需进一步的细化。

由于企业网络和应用环境的复杂性和动态性，由不同的业务目标和安全目标所得到的安全和管理策略，以及由这些策略所得到的防火墙过滤规则之间的冲突是不可避免的。另外由于防火墙过滤规则的写入是一个持续不断的过程，通常，过滤规则一旦被写入防火墙的规则库，很少会有人再去全面地审查和清理它们，随着规则的日积月累，冲突和无用的规则不可避免地导致安全受到损害以及防火墙性能的降低。防火墙过滤规则的分析和验证早已得到很多学者和研究人员的关注，并且提出了很多有用的技术和工具。但目前大多数的冲突消解依靠加入解决规则来解决冲突，虽然能够暂时解决冲突，却增加了规则的数量，使得以后的规则的添加和验证更为复杂。另外，由于规则的顺序敏感性，过滤规则的数量和组织将显著影响防火墙的规则匹配过程，为提高匹配过程的效率，降低检索时间和空间需求，必需对规则进行合理的排序和组织，并且尽量减少规则的数量[Panko2003]。因此需要能够对现有的防火墙规则库进行彻底地重写和优化，从而确保防火墙的安全和性能。

下面介绍一种空间几何建模技术，可以帮助分析人员发现和消解规则冲突，并能够对防火墙规则库进行彻底的改写和全面优化，大大减少了规则的数量，从而提高了防火墙的性能。在空间几何模型中，每个过滤规则被映射成多维空间中的一个规则几何体，从而使得防火墙配置的语义可视化。语义的可视化使冲突的鉴别和消解直观而简单，并且能够将通常定义在两个规则之间的冲突的概念和分类扩展到多个规则之间。

5．3．1 防火墙过滤规则的冲突及分类

大多数的防火墙有两个重要的性质：一是采纳了“封闭(closed)”策略，即一个信息流在缺省状态是被禁止的，除非有一条规则明确规定它是允许的；二是规则是对顺序敏感的，遵循“第一条匹配规则将适用”的原则，对于每个新的网络连接，防火墙都将从规则库中的第一条规则开始，按顺序检索直到发现一条匹配规则，尽管目前许多防火墙采用了一些优化措施，但匹配原则基本上是不变的。

在“封闭”策略的上下文中，“禁止”规则不是必需的，它们的引入主要是为了提供对例外情况的表达，以方便规则的组织。然而，“禁止”规则的引入不可避免地导致规则冲突的产生。有的规则冲突是所期望和可接受的，有的则是不可接受的，与网络安全访问策略相冲突。当规则库中过滤规则数量很大时，产生规则冲突的可能性将会非常高。

过滤规则冲突可分为以下五种类型[Al-Shaer2004]：

影子规则冲突：一个规则，如果在它前面的一个或多个规则匹配了所有能与该规则相匹配的分组，并且这些规则与该规则的动作相反，称该规则为影子规则，它永远不会被激活。影子规则冲突表明过滤规则配置出现错误，因为可能会导致合法的信息流被阻塞或非法的信息流被允许。

冗余规则冲突：一个规则，如果在它前面的一个或多个规则匹配了所有的能与该规则相匹配的分组，并且这些规则与该规则的动作相同，称该规则为冗余规则，它永远不会被激活。冗余规则也可以认为是一种错误，不过没有影子规则严重，因为它不必要地增加了规则库中规则的数量。

泛化规则冲突：一个规则，如果能够匹配与它前面的某个规则相匹配的所有分组，并且二者的动作相反，则称之为前面那个规则的泛化规则。一个规则可以是多个规则的泛化规则。被泛化的规则常用来表达一个通用规则的例外部分，因此不一定表示有错误发生。

关联规则冲突：如果两个动作相反的规则所匹配的分组的集合部分重叠而不是包含，则表示它们是关联规则，重叠部分对应信息流的允许与否取决于这两条规则的顺序关系，因此这种类型的冲突应特别小心。关联规则冲突不一定表示规则库中产生错误。

不相关规则冲突：如果一个规则的源或目标地址与该防火墙所保护的网络或通过该网络所能到达的网络都不能匹配，则该规则被认为是一条不相关规则，它是多余的应该被删除。

5．3．2 防火墙过滤规则的建模方法

每个防火墙过滤规则可以被映射成 3 维空间中的一个长方体，该 3 维空间的 3 个坐标分别代表源 IP 地址、目标 IP 地址和服务端口，每一个过滤规则在 3 维空间的映射均被赋予某种颜色，代表该规则动作字段的值，其中黑色代表“允许”，灰色代表“禁止”。

例如，下面一条过滤规则允许子网 8.0.1.* 中的机器访问由 8.0.3.* 网络提供的音频服务：

源 IP 地址	*目标 IP 地址*	*协议*	*端口*	*动作*
*8.0.1.**	*8.0.3.**	*udp*	*4000-6000*	*permit*

该规则在 3 维空间的映射是一个黑色的长方体，如图 5－16 所示。

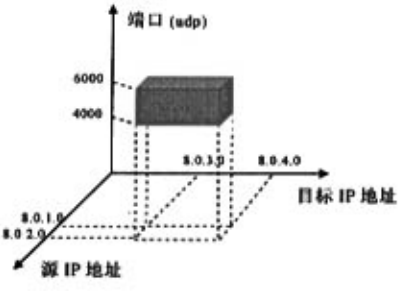


图 5－16 过滤规则在三维空间的映射

Fig. 5-16 The Mapping of the Filtering Rules to the 3-Dementional Space

尽管可以把模型空间的维数扩展到 4 维，把过滤规则的协议部分也包含进来，但本文认为这将会没有必要地增加模型的复杂度，因为一个防火墙规则库中所涉及到的协议是非常有限的，涉及不同协议的过滤规则从所要进行的分析的角度来看，可以被认为是不相关的，因而可以对它们分别进行建模和分析。模型空间维数的降低可以使模型更加简单和直观，绝大部分情况下，模型空间的维数可以进一步降低到 2 维，只剩下源和目标 IP 地址坐标，这是因为大多数过滤规则只涉及

到一些知名的服务，如 http、ftp、telnet、smtp 等等，涉及不同服务的过滤规则同样可以认为是不相关的，可以分别进行建模和分析。即使象图 5－16 所示的过滤规则使用一个连续的端口地址序列，但由于它们在端口坐标轴方向上的性质是相同的，因此只需分析与源和目标 IP 地址平行的一个剖面即可。

表 5－4 过滤规则

Tab. 5-4 The Filtering Rules

规则	源地址	目标地址	协议	端口	动作
1	8.0.1.*	*.*.*.*	tcp	80	permit
2	8.0.2.*	*.*.*.*	tcp	80	permit
3	8.0.3.0/16	*.*.*.*	tcp	80	deny
4	8.0.3.*	*.*.*.*	tcp	80	permit
5	*.*.*.*	8.0.33.16/8	tcp	80	permit
6	8.0.1.0/512	8.0.33.0/16	tcp	80	deny
7	8.0.1.*	8.0 33.*	tcp	21	permit
8	8.0.2.*	8.0.33.*	tcp	21	permit
9	8.0.2.*	*.*.*.*	tcp	21	deny
10	8.0.3.*	*.*.*.*	tcp	21	permit
11	8.0.1.0/768	8.0.33.20	tcp	21	permit
12	8 0.33.*	8.0.44.*	tcp	21	permit
13	*.*.*.*	*.*.*.*	*	*	deny

下面以图 2－6 所示实例网络中的防火墙 R1 为例，来演示如何利用空间几何建模技术来分析过滤规则，为了能够展示冲突分析和规则优化各种可能的情况，这里所采用的过滤规则实例与上一小节有所不同，如表 5－4 所示。表中有一组特殊的 IP 地址 8.0.33.*，属于“Internet”地址空间，代表该组织的某个合作机构的网络。

5．3．3 防火墙配置语义的可视化和规则冲突的鉴别

当过滤规则映射到多维空间以后，防火墙的配置便可直观地展现出来，规则冲突在映射过程中也可以很容易地被鉴别出来。

既然过滤规则的顺序决定着防火配置的语义，规则的映射也应该严格按其顺序进行。每个“允许”规则映射成一个黑色几何体，而每个“禁止”规则映射成一个灰色几何体，如果后面规则的映射与前面的规则有重叠部分，则重叠部分保持原来的颜色。图 5－17 和 5－18 分别是表 1 中与 http 和 ftp 服务有关的过滤规则的映射，为减少维数以降低复杂度，它们被分别映射到两个 2 维空间，可以看作是 3 维空间的两个剖面。

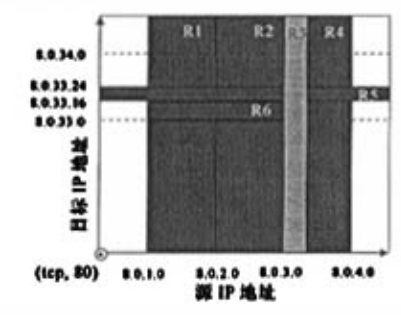


图 5－17 与 *http* 服务有关的过滤规则的映射

Fig. 5-17 Map of the Rules Concerning the *http* Service

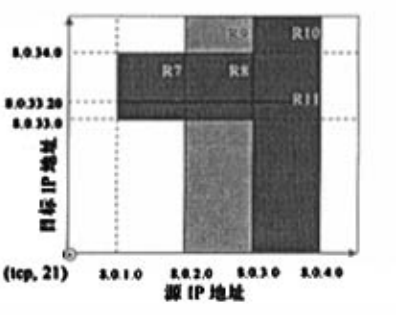


图 5－18 与 *ftp* 服务有关的过滤规则的映射

Fig. 5-18 Map of the Rules Concerning the *ftp* Service

规则冲突的鉴别可根据以下几种情况进行：

影子规则冲突：如果当前规则的映射几何体的空间已经被前面一个或多个规则的映射几何体所覆盖，并且颜色相反，则表示影子规则冲突的出现，例如在图 5-17 中，规则 6 是规则 1 和 2 的影子规则，如果想要禁止子网 DMZ 和 A 中的机器访问子网 8.0.33.0/16 提供的 http 服务，规则 6 应移到规则 1 之前。

冗余规则冲突：如果当前规则的映射几何体的空间已经被前面一个或多个的几何体所覆盖，并且颜色相同，则表示冗余规则冲突的出现。例如在图 5-18 中，规则 11 是规则 7、8 和 10 的冗余规则，它可以直接被删除。

泛化规则冲突：如果当前规则的映射几何体的空间完全包括了一个或多个已有的几何体，并且颜色相反，则该几何体是它们的泛化规则。例如在图 5-17 中，规则 4 是规则 3 的泛化规则，如果颠倒规则 3 和 4 的位置，则规则 3 成为规则 4 的影子规则，这是应该避免的。

关联规则冲突：如果两个颜色相反的映射几何体相互重叠而不是相互包含，则表示它们是关联规则，重叠部分对应的信息流允许与否取决于这两条规则的顺序关系。例如在图 5-17 中，规则 3 和 5 是关联规则。

不相关规则冲突：如果一个规则的源和目标地址与该防火墙所保护的网络或通过该网络所能到达的网络都不能匹配，即相应的分组不经过该防火墙，则该规则被认为是一条不相关规则，应该被删除。对应于多维映射空间，每一个防火墙都有一个有效空间，如果一个几何体在有效空间以外，则它对应的规则是一条不相关规则。示例防火墙的有效空间如图 5-19 所示，可以看到规则 12 在有效空间之外，则它是一条不相关规则。

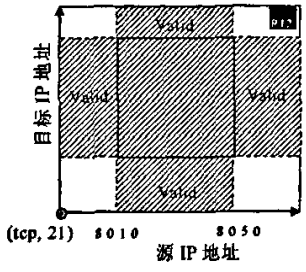


图 5－19 防火墙 R1 的有效空间

Fig. 5-19 The Valid Space of Firewall R1

5．3．4 过滤规则的重写和全面优化

当过滤规则，包括最后的缺省 “禁止(deny)” 规则，全部映射到多维空间后（缺省的“禁止”规则将剩余的空间全部填充为灰色），防火墙配置的语义便可以直观地展现给分析人员，空间中黑色的区域代表着所有被允许的信息流，管理员可以很容易地根据组织的安全策略来验证过滤规则的正确性，错误部分可以通过改变其颜色来纠正。接下来可以将规则的映射重新映射回过滤规则，以实现规则库的彻底重写。这是一个有趣而又富有挑战性的任务，它遵循以下两个原则：

1．产生的过滤规则应尽可能得少；
2．防火墙匹配分组的平均检索长度，即匹配分组所需要检索的过滤规则的平均数量应尽可能得少。

对应于规则的映射空间，第一个原则要求用尽可能少的黑色和灰色长方体（3 维空间）或矩形（2 维空间）来表示空间中的黑色区域，其中灰色几何体通常用

来表示黑色几何体中的例外部分。第二个原则要求对所产生的过滤规则进行合理的排序和组织。为降低平均检索长度，可以加入一些辅助性的“禁止”规则，这些规则不是必需的，因为最终的缺省“禁止”策略将会应用到相应的分组，但却必须检索整个规则库直至最后一条规则。增加辅助性规则虽然会增加规则的数量，但如果能找到一个平衡点的话，则是可接受的。

5．3．4．1 多维空间中黑色区域的表示和过滤规则的产生

出于简化目的，本节只讨论 2 维空间中黑色区域的表示方法。这种简化是合理可行的，因为 2 维空间可以看作是 3 维空间的一个剖面，决大多数过滤规则在端口坐标轴方向上的属性是一致的，即便不是如此，相应的过滤规则可分解为在端口方向上属性一致的多条规则，分别进行建模和分析。现在所要做的工作是如何用尽量少的黑色和灰色矩形来表示 2 维空间中的黑色区域，必须指出的是要得到最优解是比较困难的，本节的方法所得到的只是次优解。

黑色区域的表示步骤如下：

（1）在 2 维空间中每个相对独立的黑色区域由一个黑色矩形和零个或多个灰色矩形表示，其中黑色矩形沿黑色区域的边界覆盖整个黑色区域，这样可能会有灰色区域被包括进来，这些灰色区域则由一个或多个灰色矩形来表示，并且定义为该黑色矩形的“关联例外矩形(Associated Exception Rectangle)”。“关联例外矩形”的引入是必要的，当将模型映射回过滤规则时，“关联例外矩形”对应的规则必须置于被关联的黑色矩形对应的规则之前。

本节鉴别出几种具有代表性的情况，如图 5－20 所示。图中，A 本身就是一个矩形，可以直接用一个黑色矩形来表示；B 和 C 可分别由一个黑色矩形和一个灰色“关联例外矩形”来表示；D 由一个黑色矩形和两个灰色“关联例外矩形”表示。更加复杂的情况可采用同样方法处理，可能需要多于两个的灰色“关联例外矩形”来表示。

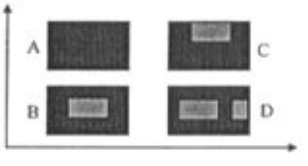


图 5－20 黑色区域的表示

Fig. 5-20 Representation of the Black Area

（2）如果出现图 5－21 所示的情况，可合并两个黑色的矩形，合并形成一个大的黑色矩形和一个灰色“关联例外矩形”。合并过程可重复进行，直至不再有黑色矩形可合并。

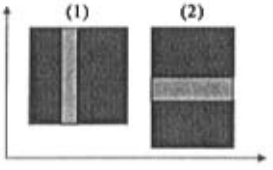


图 5－21 黑色矩形的合并

Fig. 5-21 The Combination of Black Rectangle

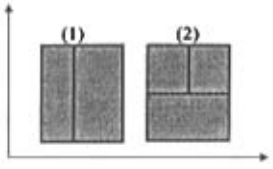


图 5－22 灰色矩形的合并

Fig. 5-22 The Combination of Black Rectangle

(3)由前面两个步骤所产生的灰色矩形也可以进行合并，两个或多个灰色矩形可合并为一个灰色矩形，在合并过程中不应产生出新的黑色矩形。图 5－22 给出具有代表性的几种合并情况，值得注意的是合并后的灰色矩形应定义为所有与被合并的灰色矩形相关联的黑色矩形的“关联例外矩形”

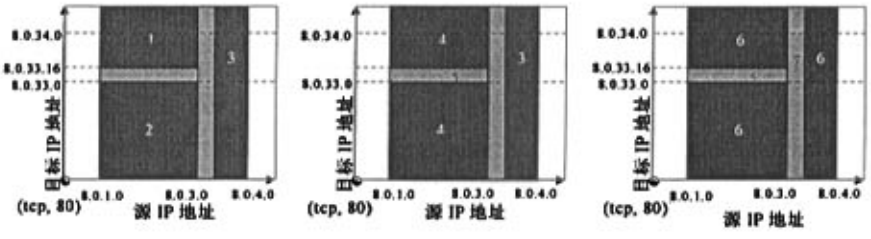


图 5－23 与 *http* 服务有关的过滤几何体的表示过程

Fig.5-23 The Representation of the Rule Objects related to *http* Service

图 5－17 中与 http 有关的过滤规则的表示过程如图 5－23 所示，图 5－23（a）所示的是原有的规则的冲突已经消解后的配置，如规则 R6 已移至规则 R1 之前，共有 3 个黑色矩形，分别标为 1、2 和 3，注意规则 R5 中与该防火墙不相关的部分已被忽略；在图 5－23（b）中，黑色矩形 1 和 2 合并成一个黑色矩形 4 和一个灰色“关联例外矩形”5；在图 5－23（c）中，黑色矩形 4 和 3 被进一步合并为一个黑色矩形 6 和一个灰色矩形 7，且灰色矩形 5 和 7 都是矩形 6 的“关

联例外矩形”。对于矩形 5、6 和 7，可以将它们映射回下面 3 个相应的过滤规则：

规则 1: (8.0.1.0/512, *.*.*.*, (tcp, 80), deny)；

规则 2: (8.0.3.0/16, *.*.*.*, (tcp, 80), deny)；

规则 3: (8.0.1.0/768, *.*.*.*, (tcp, 80), permit)。

与图 5－17 中 6 条规则相比，规则数量的减少相当可观。

5．3．4．2 过滤规则的组织和进一步优化

在一个对顺序敏感的防火墙规则库中，过滤规则的顺序不仅决定着防火墙配置的语义，而且还影响着防火墙的性能。在前面的小节中，规定每个灰色“关联例外矩形”对应的规则必须置于被关联的矩形对应的规则之前，然而无关联的矩形所对应的规则之间的顺序尚未确定下来，因为它们的排序并不影响防火墙配置的语义。本小节将给出对规则进行合理的排序和进一步优化的方法，以提高防火墙的性能。

过滤规则的排序原则是获得最短的检索长度。排序应以到达该防火墙的分组的流量统计数据为依据，适用于分组统计流量大的过滤规则应置于适用于分组流量小的规则之前。采集的流量统计样本应能反映出分组流量的真实分布。

与过滤规则对应的几何体的排序可采用下面的步骤：

（1）每一个分组流量统计样本映射为多维空间的一个带有权值的统计点，其中点的权值为相应的流量统计值。

（2）计算每个矩形所包含统计点的权值的总和。必须注意每个点只能被计算一次，如果一个统计点落入某个黑色矩形的灰色“关联例外矩形”内，则应纳入该灰色“关联例外矩形”的权值计算，尽管该黑色矩形也同样包含该统计点。

（3）根据所包含的统计点权值，按降序排列所有的矩形。

（4）最后，需要调整矩形的排序。前面曾提到灰色“关联例外矩形”必须置于被关联的矩形之前，因此如果有灰色“关联例外矩形”因为其权值小而被排在与其关联的矩形后面，则应将它移到被关联的矩形前面。

为进一步优化规则库，提高防火墙的性能，还可以加入一些辅助性的“禁止(deny)”规则，尽管它们的加入会增加过滤规则的数量，但却可以显著地降低平均检索长度。下面先给出防火墙平均检索长度 ASL 的量化的定义：

定义 5－19（平均检索长度 ASL）防火墙规则库过滤规则的平均检索长度在某个流量统计样本上计算如下：

$$ASL=1/N * \sum i * W_i \qquad i=1..n$$

其中：N 是流量统计样本总量，n 是规则库中过滤规则的数量，包括缺省的“禁止”规则，i 是规则的序列号，W_i 是第 i 条规则对应的几何体所包含的统计流量的权值。

辅助“禁止”规则的鉴别和加入过程包括以下几个步骤：

（1） 找出映射空间中不属于任何黑色或灰色矩形的具有最大权值的统计点。

（2） 以该统计点为基点，构造一个包含该点的矩形，该矩形应尽可能地延伸以包含尽可能多的统计点，但不能与任何已有的黑色矩形重叠，通常有两种选择，一个是沿源 IP 地址坐标轴扩展到最大，另一个是沿目标 IP 地址坐标轴扩展，将选择权值最大的那个，如果一样大，则可随机选择。

（3） 将新的矩形（灰色）插入到已排序的矩形列表中，值得注意的是矩形的插入应从尾部开始，这是因为经过调整后的矩形列表已不是严格按降序排列的。

（4） 计算新的 ASL’。如 ASL’<ASL,则新矩形的加入是可接受的，并重复步骤（1）－（4），否则，是不可接受的，撤消新矩形的加入，并结束本过程。

图 5－24 给出的与 ftp 服务有关的流量统计数据仅作为演示目的，并非是真实的数据。图 5－24（对应于经过矩形合并的图 5－18）中有两个黑色矩形 I 和 II，矩形 I 和 II 所包含的统计点的权值分别为 W_I= 20 + 30 = 50 和 W_{II}=50+100+120+130=400,因此矩形 II 应置于 I 前面。平均检索长度计算如下：

$$ASL=(400*1+50*2+450*3)/900=2.056$$

现在如果加入辅助灰色矩形III（包含了权值最大的不属于 I 和 II 的统计点 150），如图 5－24 所示，矩形III的权重为 W_{III}＝50＋100＋150=300，因此III应置于 I 之前、II之后。新的 ASL’计算如下：

ASL'=(400*1+300*2+50*3+150*4)/900=1 944

ASL'<ASL, 因此矩形III的添加是可接受的。如果进一步加入矩形IV，ASL 计算如下：

ASL"=(400*1+300*2+80*3+50*4+70*5)/90=1.989

ASL">ASL'，因此矩形IV的加入是不可接受的。

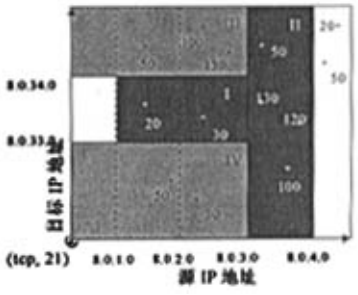


图 5-24 与 *ftp* 服务有关的流量统计

Fig.5-24 The Traffic Statistics Related to the *ftp* Service

对于防火墙过滤规则的建模和优化，目前已有学者和研究人员提出了很多值得借鉴的方法。例如，Al-shaer 和 Hamed 提供了一个过滤规则的建模方法，他们使用策略树来表示过滤规则，策略树是是一个单根的树结构，每一个从根结点到叶子结点的路径代表一个过滤规则，并给出了直观的方法来鉴别过滤规则之间的关系和冲突[Al-shaer2004]。然而，在他们的方法只能鉴别出两个规则之间的关系和冲突，事实上，如前面所述，过滤规则的冲突可能发生在多个规则之间。另外，他们的方法也无法对防火墙的规则库进行全面的优化。

5.4 小结

在策略演化过程中的各个阶段，对应于策略的各个抽象层次内部和各层次之间，都需要对策略进行各种分析和验证。由于在策略的演化过程中各个阶段所处的上下文环境不同，策略的抽象层次不同，因此，各个阶段的分析和验证的侧重点和内容也不一样，所采用的分析方法和建模工具也不尽相同。本章就本文的分析实例，分别在策略的高层、中间层和低层三个不同抽象层中各选出一到两项重要的策略分析和验证内容进行了研究。

对于高层策略的分析，本章给出了在业务流程及相关策略模板的实例化时，如何实现策略实例层次上的安全约束，如职责分离准则，以及在满足安全约束的要

求下，如何进行合理的人员和资源规划及分配等等的方法。为实现这些目标，本文使用彩色 Petri 网对业务流程、安全约束和人力资源进行建模，利用线性代数技术，通过对彩色 Petri 网的可达性分析来验证业务流程的可达授权状态；利用 Petri 网的覆盖图来计算业务流程的有效执行链，从而实现了人力资源规划。对业务流程或工作流（Workflow）的权限、安全约束和资源规划的建模和分析的方法目前得到了很多学者的重视，和本章所使用的方法最接近的是 Atluri 的工作流授权模型[Atluri2000]，他们使用彩色 Petri 网作为建模工具，同样引入了基于角色的授权(Role-based Authrization)和 SoD 的概念，但却忽略了基于角色的访问控制中的一个重要概念——角色层次结构，而且在 他们的方法中，SoD 的表达也较为复杂和有限。Atluri 只分析了可达授权状态，而没有提供人力资源规划的方法。

对于中间层策略的分析和验证，一个非常重要的任务是验证中间层策略是否满足或符合高层策略的要求，是否与高层策略相矛盾等等。本章以对网络访问控制策略的分析为例，仍采用彩色 Petri 网作为主要的分析工具，通过一个经过扩展的彩色 Petri 网框架，可在系统和抽象设备层上对网络、网络访问控制策略、网关的路由或过滤规则进行建模，利用 Petri 网的覆盖图分析网关配置的正确性。对于网络的建模和分析，研究人员一般均使用无向图[Bartal1999, Mayer2000]，这是因为实际中的网络通信通道通常可允许双向的信息流。使用 Petri 网对网络进行建模是一个新颖的方法，这是因为 Petri 网是一种有向图，其难点和关键是如何根据所要进行的分析内容，动态地确定弧的方向。

对于低层策略的分析和验证，本文选择了防火墙的过滤规则作为分析对象，这里的低层策略，即防火墙过滤规则，与现实世界中防火墙配置中的过滤规则是一一对应的，可以直接用来配置防火墙而无需进一步的细化。本章给出一种空间几何建模技术，可以帮助发现和消解过滤规则的冲突，并能够对防火墙规则库进行彻底的改写和全面优化，大大减少了规则的数量，从而提高了防火墙的性能。在空间几何模型中，每个过滤规则被映射成多维空间中的一个规则几何体，从而使得防火墙配置的语义可视化。语义的可视化使冲突的鉴别和消解直观而简单，并且将通常定义在两个规则之间的冲突概念和分类扩展到多个规则之间。在策略的分析和验证方面，空间几何建模技术一般常见于网络的 QoS 领域，用来对 QoS 策略进行覆盖性检查(Coverage Check)，以验证 QoS 策略的完全性。本章使用空间几何建模技术对过滤规则进行建模，不仅使得规则冲突的发现和消解简单化和直观化，而且为规则库的全面优化提供了一种有效的方法。

第六章 策略的翻译、发布和执行

策略翻译是跨越业务目标与底层支持技术或机制之间鸿沟的桥梁，策略翻译将业务目标最终转换成能够支持业务目标、并与底层设备或服务的配置相对应的形式，从而使得管理策略得以执行和实施[Verma2002]。策略的翻译是目前该领域尚待解决的一个难题，现有的方法和机制都存在着局限性，仅适用于各特定的应用领域，通用的或可以适用于多个应用领域的翻译技术仍亟待解决。

策略的制定和翻译一般是集中进行的，这也是基于策略的管理方法的特点和优势之一，这种集中制定和翻译可以确保管理策略的正确性、全面性和有效性，并能减轻策略自身管理任务的负担。集中制定和翻译后的策略需要分发到系统中相应的设备或用户，然后才能得以执行。策略的分发由策略分发机制来完成。

在基于策略的管理方法中，策略的执行一般由驻留在被管理设备上的代理负责或由系统管理员等负责，当接收到策略后，代理或管理员负责配置相应的设备或服务，或执行相应的动作。代理的结构取决于所采用的策略发布机制。例如，如果使用 LDAP 来发布策略，则代理必须包含一个 LDAP 客户端，此时，代理应实现 IETF 策略框架中定义的 PDP 和 PEP 两个组件的功能，因为策略库只是个目录服务器，只能由代理来完成策略的选取。相反，如果选择 SNMP 或 COPS 作为策略发布机制，则代理只需实现 PEP 的功能即可。

策略的翻译、发布和执行既可以完全手工完成，也有一部分策略可以由机器自动完成相应的处理。限于篇幅，本章不对策略翻译、发布和执行做深入的探讨，下面将简单介绍几种目前常用的策略翻译、分发和执行方法。

6．1 策略的翻译

策略的自动翻译可以采用脱机或在线两种方式进行。脱机方式的策略翻译又称静态策略翻译，使用静态的数据和预先确定下来的转换规则将高层策略转换成可应用于系统各组件的特定的策略集；在线方式的策略翻译又称实时策略翻译，使用在线组件，实时动态地监视系统元素的行为，并由翻译模块根据所观测到的动态数据动态地改变设备或服务的配置，以确保实现预定的管理目标[Beigi2004]。

目前常用的策略翻译方法包括使用静态规则、策略表查表和基于案例的推理等方法。静态规则法和策略表查表法一般用于脱机策略翻译，而基于案例的推理法可用于脱机和在线两种策略翻译方式。

1. 静态规则法(Static Rules)

使用静态规则进行策略翻译是最简单但有时却是最有效的一种方法，前提是必须事先定义有一套静态的转换规则，用来将高层管理目标转换成可被系统理解和执行的低层策略或系统配置参数。这些转换规则一般由熟悉系统细节并了解管理目标的系统管理专家来编写。该方法适用于由一组系统管理员管理系统的情况，每个管理员有着各自的职责和权限，可定义和编辑相应的策略[Beigi2004]。

2. 策略表查表法(Policy Table Lookup)

使用策略表查表法需要事先定义一张策略表，表中列出了所有可能的适合该系统的策略。管理员通过输入部分配置参数，向策略翻译模块查询相应的策略。为了实施策略转换，策略表中的每个策略应首先被映射成 N 维空间的一个对象，就像上一章分析防火墙过滤规则的方法一样，其中 N，即空间的维数，是策略中的参数的个数。每个策略对应于空间的一个区域，每个区域指向一个动作序列。策略转换系统根据需要转换的策略的查询参数定位在 N 维空间中的区域，查找能够包含该区域的、对应于策略表中某个策略的区域。 在很多情况下，需转换策略所对应的空间区域不是简单地被策略表中某一个策略对应的区域所完全包含，而是与多个策略区域相重叠，此时，应该把需转换策略对应的区域根据重叠区域分割成几个相应的部分，每个部分对应于一个新的策略。为了确保需转换策略所覆盖的区域能够被策略表中的某个或多个策略所完全包含，需要对策略表进行覆盖性检查[Beigi2004]。

3. 基于案例的推理(Case-based Reasoning)

基于案例推理的策略翻译方法利用案例数据库或系统行为历史记录作为推理的依据（已知事实），将高层策略转换为低层策略，或相反，将低层策略映射到高层策略。在该方法中，策略翻译模块根据从过去系统的行为中学到的知识来预测系统现在和将来的行为，因此系统需要维护一个案例数据库，每个案例的数据应包括相应的系统配置参数和对应于这些配置时系统所能支持或实现的目标。当需要将高层策略转换成低层策略，即将目标映射到相应的系统配置参数时，翻译模块将从案例库中寻找最合适的匹配，或进行插值计算以确定合适的配置参数。该方法的关键在于如何选择构成案例的合适的配置参数和目标，参数和目标的选择取决于各系统和应用领域特定的要求。基于案例的推理的有效性还取决于是否拥有足够的案例作为推理的依据。对于一个已经运行了一定时间的系统，案例库的

建立是可能的，而对于一个新的系统，没有系统运行历史可供参考，可使用一些实验数据或使用启发式问题来帮助获得理论数据[Beigi2004]。

6．2 策略的发布

策略发布的主要任务是将低层策略或配置信息从某个中心位置发送到系统中的各相关设备和用户，从而使策略得以执行或遵守。下面简单介绍几种常用的策略发布方法[Verma2002]。

1． 使用系统管理架构(Systems Management Framework)

在传统的网络管理中，早已有许多厂商提供了从一个控制中心对网络和系统进行管理的平台，如 IBM 的 Tivoli 系统的 TME 架构[TME1997]和 CA(Computer Association)的 TNG 架构[TNG1998]。尽管这些平台是专用的，具体的实现细节有所不同，但都有者共同的特征，即每个系统管理平台都包含一个控制台（控制中心）和多个代理，代理位于被管理设备上，负责执行由控制台发来的指令，同时也能向控制台发送报告和警报[Verma2002]。

系统管理架构通常利用中间件(Middleware)向服务器和代理提供公共服务，TME 和 TNG 提供了基于 CORBA 的通讯中间件。管理框架所提供的基本公共服务包括用户与管理平台的接口、管理节点之间以及与控制台之间的通讯等等。

使用系统管理框架发布管理策略需要在原有的框架中增加两个新的模块：一是在控制台中增加一个策略控制模块，用于策略的处理；二是在各被管理设备上增加策略代理用于接收和执行策略。策略的发布可以直接使用管理框架原有的发布机制。

使用现有的管理框架发布策略的优点在于方便实用，可充分利用现有的功能和机制。许多策略的管理和发布所需要的功能在管理框架中都有提供，如网络拓扑、设备和应用的发现、通讯机制和用户接口等等。缺点是现有的管理框架通常是为通用的操作系统设计的，如主机系统、UNIX 服务器、PC 等等，对其它设备如路由器、交换机等的支持较少，并且占用了大量的被管理设备上的资源。另外由于使用 CORBA 等机制，因而管理控制台和被管理设备之间需要使用许多动态端口建立大量的连接，使得它们在跨企业的系统中很难被采用[Verma2002]。

2． 使用脚本(Script)

许多服务器、PC、路由器等支持远程登录和管理，因此可以通过执行一个命令行脚本文件远程登录到被管理机器，并执行脚本中的命令，来完成策略的传

递。为了编写正确的脚本，策略管理工具需要了解系统中各种设备的类型，然后连接到被管理设备并调用相应的命令来配置设备。

使用脚本发布策略的最大的好处是不需要开发和安装额外的软件，缺点是管理工具必须了解所有类型的设备所提供的命令行命令及形式，并受限于设备所提供的命令行命令，因此只适用于很小一部分策略的发布和执行。

3. 使用 LDAP

为了减轻策略管理工具的负担，可以使用一个中心策略库，存放标准形式的低层策略和配置信息，管理工具将策略转换成标准形式并放入策略库中，然后各设备上的代理从策略库中取出合适的策略并执行。使用 LDAP 访问协议的目录服务器正是策略库的最佳选择，这是由于 LDAP 客户端已存在于绝大多数现有的平台上而无需重新开发新的协议，另外绝大多数的 LDAP 服务器也可以通过配置，很容易地包容各种类型的记录，包括各种策略信息。

为使用 LDAP 发布策略，策略管理工具和策略代理之间必须事先达成两个基本的约定：一是目录中策略的具体格式；二是被管理设备中的代理提取与该设备有关的策略的方法和步骤。

4. 使用 SNMP

SNMP 在网络管理中早已得到广泛的应用，每当有新的协议或技术被开发和应用到网络环境中，都无一例外地定义了相应的 MIB，因此使用 SNMP 来发布相关的策略也是一种自然而然的选择。策略管理工具将低层策略映射到相应的 MIB 值，然后通过 SNMP 的 **set** 命令来配置设备。

使用基于 SNMP 的方法发布策略的一个优点是可以以一种标准的方法应用在现有的网络环境中，缺点是安全性能差，效率低，应用范围有限。

5. 使用 COPS 服务(Common Open Policy Service)

COPS 最初用于协商资源预留策略 RSVP[Durham2000&Herzog2000]，目前该协议以被扩展，能够支持 DiffServs 和 IPSec 策略的协商和发布。使用 COPS 发布策略需要在 COPS 客户端和 COPS 服务器之间建立连接，并交换相应的信息，每个交换信息由各种类型的 COPS 对象组成，它们包含了与策略和协商有关的信息，这些对象构成了 PIB(Policy Inforamtion Base)。一个 PIB 是一个可以由路由器实施的策略规则的集合。使用 PIB 和 SNMP 中使用 MIB 的含义和方法相同，每当有新的应用或设备时，可利用已有的协议和工具，只需定义新的 PIB 即可。

使用 COPS 的优点在于它是专门为策略的发布而设计的，因而效率比 SNMP 高，但由于是新的协议，因此不像 SNMP 那样为很多设备和应用所支持。

6. 基于 Web 服务器的方法

基于 Web 服务器的方法是一个简单可行的发布策略机制，由每个设备上的代理通过基于 URL 的访问机制向策略服务器请求管理策略或配置信息。该方法的优点是可扩展的、并且能够跨越防火墙，因此适合安全策略的发布，另一个优点是代理可以是通用的，并且实现简单。缺点是策略管理工具的负担过重，它必须了解系统中所有的设备的能力和配置脚本的形式[Verma2002]。

表 6－1 中列出了上述各种策略发布机制的比较[Verma2002]。

表 6－1 现有策略发布机制的比较

Tab. 6-1 Comparation of Existent Policy Distribution Mechanism

	系统管理框架	脚 本	LADP	COPS	SNMP	Web 服务器
控制台复杂度	高	高	低	低	低	高
代理复杂度	低	无	中	中	中	低
错误控制	好	差	一般	好	好	好
延 误	低	低	中	低	低	低
标准化	专用	专用	标准	标准	标准	标准
成熟度	有高有低	高	高	低	高	高

在实践中，根据系统中具体的设备或服务，可以同时使用上述方法的几种。

6．3 策略的执行

访问请求或分组的到来等等事件都可以触发相应的策略执行机制，如访问控制机制或分组过滤机制等，去查找适用的策略以做出正确的决策。策略的匹配是策略执行中的一个关键步骤，决定了策略执行的正确性和效率，下面简单介绍几种常用的策略匹配、搜索方法[Verma2002]。

1. 线性查找法(Linear Search)

线性查找法是目前最常用的方法之一，策略根据优先级的高低或其它要求按顺序排列在线性表中，策略查找时，从第一条策略开始按顺序进行，并遵循“第一条匹配的策略将适用”的原则，正如在大多数防火墙中采用的方法一样。

线性查找法的优点是实现简单直接，当策略规则数目较少时，该方法的性能很好。但当策略数目很大时，匹配策略的时间相对较长，其时间复杂度为 O(n)，n 为策略的数目。

2. 搜索树法(Search Tree)

在本方法中，策略根据某个字段按树的搜索法查找，因此搜索的时间复杂度降为 O(log *n*)。根据策略的条件和匹配的要求的不同，策略可按照不同的方法构造成不同的树结构。例如，如果策略的匹配要求策略条件中的某个字段必须是精确匹配的，则可根据该字段的值，将策略构造成一个等分二叉树。

3. 多维搜索法(Policy Search with Multiple Dimensions)

在前一种搜索方法中，只能匹配策略的条件部分的某个字段，在大多数情况下，一个策略的条件部分包括多个字段，需要对多个字段进行匹配，如防火墙过滤规则与分组的匹配，需要对源、目标地址以及端口都要进行匹配，因此称之为多维搜索。多维搜索可采用投影搜索法，策略的每个字段都可以看作是策略在该字段坐标轴上的投影，分别对每个字段按搜索树法搜索，则可以分别得到的一个匹配策略集，各字段匹配策略集的交集中优先级最高的那个策略即是适用的策略。当然还有其它效率较高的多维搜索法，如多层表搜索法(Multitier Table Search)等[Verma2002]。

6.4 小结

策略的翻译是目前基于策略的管理领域尚待解决的一个难题，现有的方法和机制都存在着一定的局限性，仅适用于特定的应用领域。本节简单介绍了目前人们所采用的几种常见的策略翻译的方法，包括使用静态规则、策略表查表和基于案例的推理等方法。

策略发布的主要任务是将低层策略或配置信息从某个中心位置发送到系统中的各相关设备和用户，从而使策略得以执行或遵守。本节简单介绍了几种常见的策略发布方法，包括使用系统管理架构、使用脚本、使用 LDAP 或 SNMP、使用 COPS 服务、基于 Web 服务器的方法等，并对它们的优缺点做了简单的比较。

在基于策略的管理方法中，策略的执行一般由驻留在被管理设备上的代理负责或由系统管理员等负责。其中，策略的匹配是策略执行中的一个关键步骤，决定了策略执行的正确性和效率，本节简单介绍几种常用的策略匹配、搜索方法，如线性查找法、搜索树法、多维搜索法等。

第七章 结束语

本章是对本文所做的工作做一个总结。下面首先回顾和讨论本文所取得的成果，然后给出未来的研究方向。

7．1 研究成果回顾和讨论

尽管基于策略的网络管理(PBNM)的概念提出已有将近十年了，也已经取得了很多研究成果，但许多 PBNM 实践者发现在付诸实施时存在许多困难，在 PBNM 的实现过程中，包括策略的制定和实施，既复杂又耗时，而且成本偏高。另外，PBNM 的实施要求企业的组织、管理活动和资源需要相应的改动去适应技术的要求，而不是技术去适应企业管理的需要。最后，绝大多数 PBNM 解决方案仅适用于某个特定的、单一的网络环境，而无法适用于复杂的、异构的、包含多个厂商和多种产品的网络环境[Jude2001]。

在将基于策略的管理方法应用到更广泛的领域时，如本文所研究的企业的信息和系统管理，上述问题将更为突出和严重。造成上述问题的主要原因是目前的研究人员太专注于技术和细节，而没有从更宏观的范围、更高和全局性的角度去研究和解决问题，正如软件产品开发和应用时所曾经遇到的困难一样。

正是基于上述的认识，本文借鉴了软件工程中的概念、方法和技术，从全局和系统的角度去研究管理策略的整个演化过程，突出了策略自身管理的重要性，为基于策略的管理方法在企业信息和系统管理中的广泛应用指出了发展方向和奠定了坚实的基础。

在一个复杂的系统中，管理策略本身就构成了一个复杂的子系统，并形成于一个复杂而又完整的过程，因此，需要在一个系统方法学的指导下， 将管理过程中的各个环节和各种任务有机地组织起来。在研究和比较了软件工程中已有的生命周期模型后，本文以瀑布模型为基础，提出了一个合理的、完整的策略生命周期模型，明确定义了策略演化过程中各阶段的目标、任务、活动、输入/输出和参与者，以及各阶段之间的依赖关系等等，可用于指导整个基于策略的管理过程，将管理过程中的各个环节、各种管理活动和资源有机地组织起来，以确保管理策略的完整性、完全性、正确性、可实施和可维护性。

编写和定义明确的、全面的、精确和完整的系统管理策略对于成功的企业信息资源的管理是至关重要的。而信息系统的管理需求分析正是编写和定义管理策略

的依据和基础，在成功的系统管理中起着决定性的作用。 本文通过对软件工程中各种系统需求分析技术的研究和比较，采纳了基于目标的需求分析技术，在该方法的基础上，进行了相应的改进和扩展，使之能够很好地应用于信息资源管理需求分析和管理策略的定义。本文提供了基于目标的企业信息和系统管理需求分析框架模型，给出了建立目标模型的步骤和方法，以及根据目标模型定义信息和系统管理策略的规则。目标障碍分析和处理也是基于目标的需求分析中的一项重要任务，本文同样给出了目标障碍的识别和处理的方法，除采用传统的障碍处理方法外，还提供了直接根据障碍来定义相应的管理策略的方法。另外本文还提供了在系统安全管理中，如何根据恶意目标建立反目标模型，以及如何根据反目标模型确定安全需求和定义安全策略等的方法。

策略的分析和验证也是策略演化过程中的另外一个重要任务。由于在策略的演化过程中各个阶段所处的上下文环境不同，策略的抽象层次不同，因此，各个阶段的分析和验证的侧重点和任务也不一样，所采用的分析方法和建模工具也不尽相同。本文分别在策略的高层、中间层和低层三个不同抽象层中各选出一到两项重要的策略分析和验证内容进行了讨论。

对于高层策略的分析，本文给出了在业务流程及相关的策略模板的实例化时，如何实现策略实例层次上的安全约束，如职责分离准则；如何在满足安全约束的要求下，进行合理的人员和资源规划及分配等等的方法。本文以彩色 Petri 网为工具，对业务流程、安全约束和人力资源进行建模，利用线性代数技术，通过对彩色 Petri 网的可达性分析来验证业务流程的可达授权状态；利用 Petri 网的覆盖图来计算业务流程的有效执行链。

对于中间层策略的分析和验证，一个非常重要的任务是验证中间层策略是否满足或符合高层策略的要求，是否与高层策略相矛盾等等。本文以对网络访问控制策略的分析为例，仍采用彩色 Petri 网作为主要的分析工具，提供了一个经过扩展的彩色 Petri 网框架，可在系统和抽象设备层上对网络、网络访问控制策略、网关的路由或过滤规则进行建模，并利用 Petri 网的覆盖图分析网关配置的正确性。

对于低层策略的分析和验证，本文选择了防火墙的过滤规则作为分析对象，它们与现实中防火墙配置中的过滤规则是一一对应的，可以直接用来配置防火墙而无需进一步的细化。本文提供了一种空间几何建模技术，将每个过滤规则映射成多维空间中的一个规则几何体，从而使得防火墙配置的语义可视化。语义的可视化使冲突的鉴别和消解直观而简单，并且能够将通常定义在两个规则之间的冲突

的概念和分类扩展到多个规则之间。同时，在该模型的基础上，本文还提供了对防火墙规则库进行彻底的改写和全面优化的方法。

本文的主要创新性研究工作可以概括如下：

- 系统地研究了信息和系统的管理策略的生命周期，首次明确提出了策略工程和策略生命周期模型的概念，并给出了一个合理的、完整的策略生命周期模型，可用以指导实施基于策略的管理方法在企业的信息和信息系统管理中的应用；

- 在策略工程概念的指导下，明确了管理需求分析在策略生命周期中的重要性，并在借鉴了已有的需求分析技术基础上，提供了一个基于目标的企业信息和系统管理需求分析框架模型，使之能够很好地应用于信息管理需求分析和管理策略的确定，并给出了根据需求确定管理目标和管理策略的方法；

- 对于策略的分析和验证，给出了一些新的建模和分析方法，例如采用彩色 Petri 网对业务过程的授权和职责进行建模，不仅能够包含权限和职责管理，还能够包含安全约束如 SoD 以及角色层次关系结构等概念。对于网络访问控制，创新性地采用彩色 Petri 网对网络和相关访问控制策略进行建模，给出了如何根据所要进行的分析类型确定 Petri 网中弧方向的方法。在防火墙过滤规则的分析中，利用空间几何建模技术对过滤规则进行建模，将原有的两个规则之间的冲突概念扩展到多个规则之间，并给出了规则库的彻底重写和全面优化的方法。

7．2 未来研究方向

信息和系统管理策略的生命周期的研究是一个崭新的课题，本文的研究工作只是一个初步的探索，限于时间和精力，许多重要的课题本文并未涉及，下面列出其中关键和迫切的课题，作为未来的研究重点。

- 进一步完善策略工程的概念和策略的生命周期模型，将该领域的最新研究成果纳入到策略工程的框架中来，同时也可以使用策略工程的概念和观点来指导基于策略的管理方法的研究和应用。将本文的方法应用到一个实际的、大型的、复杂的系统，以检验相关概念和模型的正确性和可实施性，同时也可以发现问题和不足。

- 策略的翻译和细化是目前该领域的一个难点，缺乏实用的策略翻译方法和工具是目前基于策略的管理方法在实际应用中的一个主要障碍。尽管获得通用的和完全自动化的翻译方法和机制是十分困难的，仍需要研究人员的不懈的努力，但在朝这个方向的努力中，获得某个特定领域的，某种程度上的自动翻译是可行的[Albuquerque2005]，也是下一步的研究重点。

- 在基于目标的系统管理需求分析技术研究领域仍存在许多值得进一步研究和深化的地方，如目标和障碍的识别技术、目标的细化方法、需求的获取和相应策略的制定方法等等，都有待进一步的完善和实践检验。

- 策略的管理工具和管理平台是在实际应用中，策略的定义和实施过程中必不可少的一个支持，也是下一步将本文理论和方法付诸实践时首先要解决的一个问题之一。策略的分析和推理技术和工具也需要开发和完善，其中，对策略的执行过程和结果的仿真分析工具将是一个有趣和富有挑战性的课题。

- 随着互联网技术的发展，越来越多的企业间的业务往来和信息交流都通过网络来完成，Extranet，虚拟企业，动态联合企业等概念也应运而生，每个企业组织都有自己的策略要求和机制，而且企业间信任是有一定限度的，企业之间的业务关系也是动态变化的，因此，企业间策略的协调和管理也是策略管理中的一个新任务[Patz2001]。

参考文献

[Albuquerque2005] J. P. Albuquerque, H. Krumm, and P. L. Geus, Policy Modeling and Refinement for Network Security Systems, IEEE/IFIP International Symposium on Integrated Network Management (IM 2005). MAY 2005, Nice France.

[Alexander2002] I. Alexander, Misuse Cases, Proc. Of International Requirements Engineering Conf. (RE'02), Germany, 2002.

[Alexander2003] I Alexander, Misuse Case: Use Case with Hostile Intent, IEEE Soft\ware, 2003.

[Al-Shaer2004] E. S. Al-Shaer and H. H. Hamed,. Modeling and Management of Firewall Policies. In IEEE Transactions on Network and Service Management, Vol. 1(1), April 2004.

[Atluri2000] V. Atluri and W-K. Huang, A Petri Net Based Safety Analysis of Workflow Authorization Models, *Journal of Computer Security*, Vol.8, No. 2/3, pp. 209-240, 2000.

[Anderson1997] J. Anderson, S. Brand, L. Gong, T. Haigh, S. Lipner, T. Lunt, R. Nelson, W. Neugent, H Orman, M. Ranum, R. Schell, and E.. Spafford. Firewalls: An Expert Roundtable. IEEE Software 14 (5): 60-66,1997.

[Antón1996] A. I. Antón. Goal-Based Requirements Analysis , Second IEEE International Conference on Requirements Engineering (ICRE '96), Colorado Springs, Colorado, pp. 136-144, 15-18 April 1996.

[Antón1997] A. I. Antón. Goal Identification and Refinement in the Specification of Software-Based Information Systems, Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, GA, 1997.

[Antón1998] A. I. Anton and C. Potts, The Use of Goals to Surface Requirements for Evolving Systems, *Proc. ICSE-98: 20th Intrnational Conference on Software Enginering*, Kyoto, April 1998.

[Ahn1999] G. J. Ahn, and R. Sandhu (1999). *The RSL99 Language for Role-Based Separation of Duty Constraints*. In Proceedings of the Fourth ACM Workshop on Role-Based Access Control, Fairfax, Virginia, USA, ACM Press, pp. 43-54, 28-29 October 1999.

[Anderson2001] Ross J. Anderson. Security Engineering — A Guide to Building Dependable Distributed Systems. John Wiley & Sons, 2001.

[Atluri1996] V. Atluri and W-K. Huang, An Authorization Model for Workflows, In Proc. of the Fifth European Symposium on Research in Computer Security, pp. 44-64, September 1996.

[Atluri2000] V. Atluri and W-K. Huang, A Petri Net Based Safety Analysis of Workflow Authorization Models, Journal of Computer Security, Vol.8 (2/3): pp. 209-240, 2000.

[Atluri1998] N. R. Adam, V. Atluri, and W-K. Huang, Modeling and Analysis of Workflows Using Petri Nets, Journal of Intelligent Information Systems, Vol.10 (2): pp. 131-158, 1998.

[Bandara2005] A. Bandara, E. Lupu, A. Russo, M. Sloman, P. Flegas, M. Charalambides, and G. Pavlou, Policy Refinement for DiffServ Quality of Service Management, IEEE/IFIP International Symposium on Integrated Network Management (IM 2005). MAY 2005, Nice France.

[Bartal1999] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A Novel Firewall Management Toolkit. In Pro. Of 20th IEEE Sym. On Security and Privacy, OakLand, CA, pp.17-31, 1999.

[Baskerville1993] [Baskerville 1993] Baskerville, R, Information Systems Security Design Methods: Implications for Information Systems Development, *ACM Computmg Surveys* of December 1993 25(4): 375-414.

[Beigi2004] M. S. Beigi, S. Calo and D. Verma, Policy Transformation Techniques in Policy-based Systems Management, In Proc. of 5th IEEE Workshop on Policies for Distributed Systems and Network(Policy 2004), IBM TJ Watson Research Center, New York, USA, June 2004.

[Bell1973a] D.E. Bell, L.J. Lapadula, Secure Computer Systems: Mathematical Foundations, Technical Report ESD-TR-278, Vol.1, The Mitre Corp., Bedford, MA. 1973.

[Bell1973b] D.E. Bell, L.J. LaPadula, Secure Computer Systems: Unified exposition and Multics Interpretaion, Technical Report ESD-TR-278, Vol.4, The Mitre Corp., Bedford, MA. 1973.

[Bertino1998] E. Bertino et al, An Access Model Supporting Periodicity Constraints and Temporal Reasoning, ACM Trans. On Database Systems, Vol. 23(3), pp.231-285, Sep.1998.

[Biba1977] K.J. Biba, Integrity Considerations for Secure Computer Systems, Technical Report ESD-TR-3153, Vol.3, The Mitre Corp., Bedford, MA. Apr.1977.

[Brose2001] G Brose et al, Integrating Security Policy Design into Software Development Process, Technical Report B-01-06, Freie University, Berlin, Germany, Nov. 2001.

[BSI1999] BSI: British Standards Institute. Information Security Management —Part 1: Code of practice for information security management, 1999.

[Burns2001] J. Burns, A. Cheng, P. Gurung and D. Martin, Automatic Management of Network Security Policy, IEEE Press, 2001.

[Chivers2004] H. Chivers, Security and Systems Engineering,

[C&W1987] D.D. Clark, D.R. Wilson, A Comparison of Commercial and Military Computer Security Policies, In Proc. Of the IEEE Symposium on Security and Privacy, 1987.

[Cockburn2001] A. Cockburn, Writing Effective Use Case, Addison-Wesley, 2001.

[Crook2002] R. Crook, D. Ince, L. Lin, and B. Nuseibeh, Security Requirements Engineering: When Anti-Requirements Hit the Fan, Proceedings of IEEE International Requirements Engineering Conference (RE'02), Essen, Germany, September 2002.

[Cysneiros2001] Cysneiros, L. M. and J. C. S. d. P. Leite. *Using UML to reflect non-functional requirements.* In Proceedings of *The 2001 conference of the Centre for Advanced Studies on Collaborative research*, Toronto, Ontario, Canada. IBM Press. 2001

[Daminanou2000] N. Damianou, N. Dulay, E. Lupu, M. Sloman, Ponder: A Language for Specifying Security and Management Policies for Distributed Systems, The Language Specification—version 2.2, Research Report Doc 2000/1, Imperial College of Science Technology and Medicine, Department of Computing, London, Apr. 2000.

[Daminanou2001] N. Damianou et al, The Ponder Specification Language, Proc. Of Policy 2001: Workshop on Distributed Systems and Network, Bristol, U.K., Jan. 2001.

[Daminanou2002] N.C. Daminanou, A Policy Framework for Management of Distributed Systems, PhD. Thesis, London, Feb. 2002.

[Dardenne1993] A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal-Directed Requirements Acquisition", *Science of Computer Programming*, Vol. 20, 1993, 3-50.

[Darimont1996] R. and A van Lamsweerde, "Formal Refinement Patterns for Goal-Driven Requirements Elaboration", *Proc FSE'4 - Fourth ACM SIGSOFT Symp on the Foundations of Software Engineering*, San Francisco, October 1996, 179-190

[Darimont1998] R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde, "GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering", *Proc. ICSE'98 - 20th Intl. Conf. on Software Engineering*, Kyoto, April 1998, vol. 2, 58-62. (Earlier and shorter version found in *Proc. ICSE'97 - 19th Intl. Conf. on Software Engineering*, Boston, May 1997, 612-613.)

[DEN] The Directory Enabled Networking, Ad Hoc working Group, http://www.murchiso.com/den.

[Devanbu2000] P.T. Devanbu, S Stubblebine, Software Engineering for Security: A Roadmap, In A. Frinkelstein, editor, The Future of Software Engineering , ACM Press, 2000.

[DMTF] DMTF, Common Information Model (CIM) Schema, version 2.2, http://www.dmtf.org/spe/cims.html, June 1999.

[DoD1985] Department of Defense, Trusted Computer System Evaluation Criteria, National Security Center, DoD 5200.28-STD, Dec. 1985.

[Dulay2001] D. Dulay et al, A Policy Deployment Model for the Ponder Language, Proc. Of the 7th IFIP/IEEE International Symposium on Integrated Network Management, Seattle, USA, May. 2001.

[Durham2000] D. Durham et al., The COPS (Common Open Policy Service) Protocol, RFC 2748, Jan.2000.

[Epstein2001] Pete Epstein and Ravi Sandhu, Engineering of Role Permission Assignments, In Proceedings of 17th Computer Security Applications Conference, New Orleans, Louisiana, December 10-14, 2001.

[Ferraiolo1992] D. Ferraiolo, R. Kuhn. Role-based Access Controls, In Proc. Of the 15th NIST-NCSC National Computer Security Conference, pp. 554-563, Baltimore, MD, Oct. 1992.

[Graham1972] G.S. Graham, P.J. Denning, Protection-Principles and Practice, In AFIPS Press, editor, Proc. Spring Jt. Computer Conference, Vol. 40, pp. 417-429, Montvale, NJ., 1972.

[Gries1981] D. Gries, *The Science of Programming*. Springer-Verlag, 1981.

[Goh1998] C. Goh, Policy Management Requirements,

[Harrison1976] M.H. Harrison, W.L. Ruzzo, J.D.Ullman, Protection in Operating Systems, Communication of the ACM, 19(8): 461-471, 1976.

[Hayton1998] R. J. Hayton, J. M. Bacon, and Moody, Access Control in an Open Distributed Environment, In Proc. of the IEEE Symposium on Security and Privacy, Oakland, California, USA, May 1997.

[Heiler1996] K. Heiler and R. Wies, Policy-Driven Configuration Management of Network Devices, Proceedings of the IFIP/ IEEE Network Operations & Management Symposium, Kyoto Japan, April, pp. 674-689, 1996.

[Henning1999] R.R. Henning, Security Service Level Agreement: Qualitifiable Security for the Enterprise, In Proc. of New Security Paradigms Workshop, Caledon Hills, Ontario Canada. ACM Press, 1999.

[Herzberg2000] Herzberg, A., Y. Mass, J. Michaeli, D. Naor and Y. Ravid (2000). *Access Control Meets Public Key Infrastructure, or: Assigning Roles to Strangers*. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, USA, 14-17 May 2000.

[Herzog2000] S. Herzog, et al, COPS Usage for RSVP, RFC 2749, Jan. 2000.

[Hitchens2001] Hitchens, M. and V. Varadharajan, *Elements of a Language for Role Based Access Control*. In Proceedings of the Information Security for Global Information Infrastructures, IFIP TC11 Sixteenth Annual Working Conference on Information Security, Beijing, China, Kluwer, pp. 371-380, 22-24 August 2000.

[Knorr2001] K. Knorr and H. Weidner, Analyzing Separation of Duties in Petri Net Workflows, Lecture Notes in Computer Science, Information Assurance in Computer, Springer-Verlag Heidelberg, Volume 2052 / 2001, pp. 102-114, January 2001.

[Kruchten2000] P. Kruchten, The Rational Unified Process—and Introduction, Addison-Wesley, 2000.

[Kulak2000] D. Kulak and E. Guiney, Use Case: Requirements in Context, ACM Press, 2000.

[Hyland1998] P.C. Hyland, R. Sandhu, Management of Network Security Application, In Proc. Of the 21st National Information Systems Security Conference (NISSC), Hyatt Regency Crystal City, Virginia, USA, Oct. 1998.

[IETF2001a] IETF Policy Framework Working Group: http://www.ietf.org/html.charters/policy-charter.html.

[IETF2001b] IETF Working Group on IP Security Policy: http://www.ietf.org/html.charters/ipsp-charter.html.

[Irvine2000] C. Irvine and T. Levin, Quality of Security Service, In Proc. of the 2000 Workshop on New Security Paradigms, Ballycotton, County Cork, Ireland, 2000.

[Jacobson1999] I. Jacobson, G. Booch, and J. Rumbaugh: The unified Software Development Process, Addison Wesley, 1999.

[Jajodia1997] S. Jajodia, P. Sarmarati, and V. S. Subrahmanian, A Logical Language for Expressing Authorizations, In Proc. of the IEEE Symposium on Security and Privacy, Oakland, California, USA, pp. 31-42, May 1997.

[Jensen1992] K. Jensen, Colored Petri Nets — Basic Concepts, Analysis Methods and Practical Use, Volume 1, EATCS Monographs on Theoretical Computer Science, Springer, 1992.

[Jude2001] M. Jude, Policy-based Management: Beyond The Hype, In Business Communications Review, available from http://www.bcr.com/bcrmag/2001/03/p52.asp, pp. 403-430, 2001.

[Konstantinous1999] A. V. Konstantinous, S.Bhatt, Y.Yemini, S. Rajagopalan, Managing Security in Dynamic Networks, Proc. Of 19th USENIX System Administration Conference (Lisa'99), Seattle, WA, USA, Nov.1999.

[Lamsweerde1995] A. van Lamsweerde, R. Darimont, and Ph. Massonet, "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt", *Proc. RE'95 - 2nd Intl. IEEE Symp. on Requirements Engineering*, March 1995, 194-203.

[Lamsweerde2000] A. van Lamsweerde and E. Letier, Handling Obstacles in Goal-driven Requirements Engineering, IEEF Transactions on Software Engineering, Special Issue on Exception Handling, 2000.

[Lamsweerde2001] A. van Lamsweerde , "Goal-Oriented Requirements Engineering: A Guided Tour", *Invited Minitutorial, Proc RE'01 - 5th Intl Symp. Requirements Engineering*, Toronto, August 2001, pp. 249-263.

[Lamsweerde2004] A. van Lamsweerde, Elaborating Security Requirements by Construction of Intentional Anti-Models, , *Proc. ICSE'04 - 26th Intl. Conf on Software Engineering*, Endiburg, ACM_IEEE, IEEE CS Press, May 2004, 148-157.

[Lee2002] Y. Lee, J. Lee, and Z. Lee, Integrating Software Lifecycle Process Standards with Security Engineering, Computer & Security, Elsevier Science Publication, Vol 21, No 4, pp. 345-355, 2002.

[Letier2001] E. Letier, *Reasoning about Agents in Goal-Oriented Requirements Engineering*. Ph. D. Thesis, University of Louvain, May 2001; *http://www info ucl ac.be/people/eletier/thesis.html*.

[Letier2002] E. Letier and A. van Lamsweerde, "Deriving Operational Software Specification from System Goals", *Proc. ICSE'02 - 24th Intl. Conf. on Software Engineering*, Orlando, ACM Press, May 2002.

[Lobo1999] Lobo, J., R. Bhatia and S. Naqvi, *A Policy Description Language*. In Proceedings of the Sixteenth National Conference on Artificial Intelligence Eleventh Innovative Applications of AI Conference, Orlando, Florida, USA, 18-22 July 1999.

[Mayer2000] A. Mayer, A. Wool, and E. Ziskind. Fang: A Firewall Analysis Engine. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, 2000.

[McDermott1999] J. McDermott, and C. Fox, Using Abuse Case Models for Security Requirements Analysis, Proc. Of 15th IEEE Annual Computer Security Application Conf. 1999.

[Merill2001] D. C. Merrill, A. MacWillson, and G. Loveland, Requirements for a true Enterprise Wide Security Infrastructure: The play's the thing,

[Moffett1999] J. D. Moffett, Requirements and Policies, *Proc. Policy Workshop* HP-Laboratories Bristol, UK, 1999.

[Moffett2003] Moffett, J. D. and B. A. Nuseibeh, *A Framework for Security Requirements Engineering*, University of York, Department of Computer Science, YCS-2003-368. 20 August 2003.

[Moore2001] B. Moore et al, Policy Core Information Model—version 1 specification, RFC 3060, Feb. 2001.

[Murata 1989] Tadao Murata, Petri nets: Properties, analysis and applications[C]. Proceedings of the IEEE, April 1989, Vol.77(4): pp.541-580.

[NIST1995] *An Introduction to Computer Security: The NIST Handbook*, National Institute of Standards and Technology (NIST), SP 800-12. October 1995.at http://csrc.nist.gov/publications/

nistpubs/800-12/

[NIST2001] Gary Stoneburner, Clark Hayden, and Alexis Feringa, Engineering Principles for Information Technology Security (A Baseline for Achieving Security), NIST Special Publication 800-27, June 2001.

[OASIS2001] Organization for the Advancement of Structured Information Standards, *XACML language proposal, version 0 8, available from http //www.oasis-open org/committees/xacml*, 2001.

[OCTAVE2003] Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE), Carnegie Mellon, Software Engineering Institute, CERT Coordination Centre.at http://www.cert.org/octave/, 2003.

[OMG1999] Object Management Group, OMG Unified Modeling Language Specification, version 1.3, June 1999.

[Osborn2000] S. Osborn, R. Sandhu, Q. Munawer, Configuring Role-based Access Control to Enforce Mandatory and Discretionary Access Control Policies, ACM Trans. On Information and System Security, Vol. 3(2), pp. 85-106, Mat. 2000.

[OSI1994] A. Langsford, OSI Management Model and Standards, Chapter4, in Network and Distributed Systems Management, pp. 69-93, 1994.

[Panko2003] R. Panko. Corporate Computer and Network Security, Prentice Hall, 2003.

[Patz2001] G. Patz et al, Multidimensional Security Policy Management for Dynamic Coalitions, IEEE Network, 2001.

[Permpoontanalarp2001] Y. Permpoontanalarp and C. Rujimethabhas. A Unified Methodology for Verification and Synthesis of Firewall Configurations. ICICS 2001, LNCS 2229, Springer-Verlag Berlin Heidelberg, pp. 328-339, 2001.

[Potts1995] C., Using Schematic Scenarios to Understand User Needs, *Proc. DIS'95 - ACM Symposium on Designing interactive Systems: Processes, Practices and Techniques*, University of Michigan, August 1995.

[RFC3198] A. Westerinen et al, Terminology for Policy-based Management, RFC3198, Nov. 2001.

[Ribeiro2001] Ribeiro, C., A. Zuquete and P. Ferreira, *SPL: An access control language for security policies with complex constraints*. In Proceedings of the Network and Distributed System Security Symposium(NDSS'01), San Diego, California, February 2001.

[Sandia2005] Bay Area Rapid Transit District, Advance Automated Train Control System, Case Study Description. Sandia National Labs, http:// www.hcecs.sandia.gov/bart.htm, visited at May 2005.

[Samarati2000] P. Samarati, S. Vimercati, Access Control: Policies, Models, and Mechanism, In Foundation of Security Analysis and Design (Tutorial Lectures), R. Focardi and R. Gorrieri eds., Springer-Verlag, pp.137-196, Sep. 2000.

[Sandhu1990] R. Sandhu, S. Jajoda, Integrity Mechanism In Database Management Systems, In Proc. Of the 18th NIST-NCSC National Computer Security Conference, Washington D.C., pp. 526-540, Oct.1990.

[Sandhu1993] R. Sandhu, On Five Definition of Data Integrity, In Proc. of the IFIP WG 11.3 workshop on Database Security, Lake Guntersville, Alabama, Sep. 1993.

[Sandhu1996] R.S. Sandhu, E.J. Coyne, H. L. Feinstein, C.E. Youman, Role-based Access Control Models, IEEE Computer, 29(2): 38-47, Feb. 1996.

[Sandhu1998] R S. Sandhu, Q.Munawer, How to do discretionary access control using roles, In Proc. of 3rd ACM workshop on Role-based Access Control (RBAC'98, Fairfax), ACM Press, New York, NY, Oct. 1998.

[Schach1998] S., Software Engineering with JAVA, forth edition, the McGraw-Hill Companies, Inc., 1998.

[Schneier2003] B., *Beyond Fear: Thinking Sensibly About Security in an Uncertain World.* 2003: Copernicus Books 0387026207.

[Sindre2000] G. Sindre, A. l Opdahl, Eliciting Security Requirements by Misuse Cases, Proc. of 37th Conf. Techniques of Object-Oriented Language and Systems, TOOLS Pacific 2000, pp. 120-131, 2000.

[Sindre2001] G. Sindre, and A. l. Opdahl, Templates for Misuse Case Description, Proc. 7th International Workshop on Requirement Engineering, Foundation for Software Quality (REFSQ'2001), June 2001.

[Sloman1993] M.S. Sloman, J. Magee, K. Twidle, J. Kramer, An Architecture for Managing Distributed Systems, In proc. 4th IEEE workshop on Future Trends of Distributed Computing Systems, Lisbon,Portugal, IEEE Computer Society Press, pp. 40-46, Sep.1993.

[Sloman1994] M.S Sloman, Policy Driven Management for Distributed Systems, Journal of Network and systems Management, Vol. 2(4), pp. 333-360, Dec. 1994.

[SP2002] Security-Patterns Web-Site, http://www.security-patterns.de, 07/24/2002, 2002.

[Schumacher2001] Markus Schumacher and Utz Roedig, Security Engineering with Patterns. In Proceedings of the 8th Conference on Pattern Languages of Programs (PLoP 2001), September 2001, http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/SR01-1.html.

[SSECMM1999] SSE-CMM Project, Systems SE Capability Maturity Model-version 2.0, April 1999, http://www.sse-cmm.org/Papers/SSECMMv2Final.pdf, 1999.

[Stevens1994] W. R. Stevens, TCP/IP Illustrated, Volume 1: The Protocols. Addsion-Wesley, 1994.

[Strassner2002a] J. Strassner, How Policy Empowers Business-Driven Device Management, Proc. Of 3rd IEEE International workshop on Policies for Distributed Systems and Networks (Policy'02), 2002.

[Strassner2002b] J. Strassner, DEN-ng: Achieving Business-Driven Network Management, IEEE network, 2002.

[Taegen1995] T. Taegen, A. Prakash, Requirements of Role-based Access Control for Collaborative Systems, In Proc. Of the 1st ACM workshop on Role-based Access Control, Gaithersburg, MD. USA, Nov. 1995.

[Tettero1997] O. Tettero, D.J. Franken, and J. Schot, Information Security Embedded in the Design of Telematics Systems, Computer & Security. Vol 16(2), pp. 145-164, 1997.

[Thomen1998] Dan Thomsen, Dick O'Brien, Jessica Bogle. Role Based Access Control Framework for Network Enterprises, In Proceedings of 14th Annual Computer Security Application Conference, December 7-11, 1998.

[Thomas1994] R.K. Thomas, R.S. Sandhu, Conceptual Foundations for A Model of Task-based Authorizations, Proc. Of the IEEE Computer Foundation workshop, New Hampshire, 1994.

[Thomas1997]R.K. Thomas, R.S. Sandhu, Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management, Proc. of the IFIP WG11.3 workshop on Database Security, Lake Tahoe, CA, Aug. 1997.

[TME1997] R. Lendernmann, An Introduction to Tivoli's TME 10, Prentice Hall, Oct. 1997.

[TNG1998] R. Sturm, Working with Unicenter TNG, Que Education and Training, Aug. 1998.

[Verma2001] D.C. Verma, Policy-base Networking: Architecture and Algorithm, New Riders Publishing, 2001.

[Wies1994] R.Wies, Policy Definition and Classification: Aspects, Criteria, and Examples, Proc. O f the IEIP/IEEE International workshop on Distributed Systems: Operation & Management, Toulouse, France, Oct.1994.

[Yoder1997] J. Yoder and J. Barcalow. Architectural patterns for enabling application security. In Proceedings of the 4th Conference on Patterns Language of Programming (PLoP'97), September 1997. citeseer.nj.nec.com/yoder98architectural.html.

[科飞 2002] 科飞管理咨询公司，信息安全管理概论—BS7799 理解与实施，机械工业出版社，2002.7.

附录 A. KAOS 提供的目标精化、障碍和操作化模式(部分)

A.1 目标精化模式 (Goal Refinement Patterns)

模式	目标	子目标 1	子目标 2	子目标 3
RP1	$P \Rightarrow \Diamond Q$	$P \wedge R \Rightarrow \Diamond Q$	$P \Rightarrow \Diamond R$	$P \Rightarrow P \mathcal{W} R$
RP2	$P \wedge R \Rightarrow (P \wedge R) \mathcal{W} R$	$P \Rightarrow P \mathcal{W} R$	$Q \Rightarrow Q \mathcal{W} R$	
RP3	$P \Rightarrow \Diamond Q$	$P \Rightarrow \Diamond R$	$R \Rightarrow \Diamond Q$	
RP4	$P \Rightarrow \Diamond Q$	$P \wedge P1 \Rightarrow \Diamond Q1$	$P1 \wedge P2 \Rightarrow \Diamond Q2$	$\Box (P1 \vee P2)$, $Q1 \vee Q2 \Rightarrow \Diamond Q$
RP5	$P \Rightarrow \Diamond Q$	$P \wedge \neg R \Rightarrow \Diamond R$	$P \wedge R \Rightarrow \Diamond Q$	$P \Rightarrow \Box P$
RP6	$P \Rightarrow \Diamond Q$	$\neg R \Rightarrow \Diamond R$	$P \wedge R \Rightarrow \Diamond Q$	$P \Rightarrow \Box P$
RP7	$P \Rightarrow \Diamond Q$	$P \Rightarrow \Diamond R$	$R \Rightarrow R \mathcal{U} Q$	

A.2 障碍模式 (Patterns of Obstacles to Goals)

模式	目标	域属性	障碍
1－step 回溯	$R \Rightarrow \Diamond S$	$S \Rightarrow P$	$\Diamond (R \wedge \Box \neg P)$
	$R \Rightarrow \Diamond S$	$S \Rightarrow P$	$\Diamond (R \wedge (\neg S \mathcal{U} \Box \neg P))$
	$P \Rightarrow \Box Q$	$Q \Rightarrow C$	$\Diamond (P \wedge \Diamond \neg C)$
饥饿法	$R \Rightarrow \Diamond S$	$S \Rightarrow P$	$\Diamond (R \wedge \Box (\neg S \mathcal{U} \Box \neg P)$
非持续性	$R \Rightarrow \Diamond S$	$R \wedge \Diamond S \Rightarrow P \mathcal{W} S$	$\Diamond (R \wedge \neg S \mathcal{U} (\neg P \wedge \neg$
里程碑	$R \Rightarrow \Diamond S$	$R \wedge \Diamond S \Rightarrow \neg S \mathcal{W} M$	$\Diamond (R \wedge \Box \neg M)$
阻塞	$R \Rightarrow \Diamond S$	$B \Rightarrow \Box \neg S$	$\Diamond (R \wedge (\neg S \mathcal{U} \neg B))$
替换法	$R \Rightarrow \Diamond S$	$S' \Rightarrow \Box \neg S \wedge \blacksquare \neg S$	$\Diamond (R \wedge \Diamond S')$
强化法	$R \Rightarrow \Diamond S$	$R \wedge \Diamond S \Rightarrow \Diamond (P \wedge (P \mathcal{W} S))$	$\Diamond (R \wedge \Box \neg P)$
回推法	$P \Rightarrow \Box Q$	$C \Rightarrow \Diamond \neg Q$	$\Diamond (P \wedge \Diamond C)$
1－state back	$P \Rightarrow \Box Q$	$C \Rightarrow \circ \neg Q$	$\Diamond (P \wedge \Diamond C)$

A.3 目标操作化模式(Patterns of Goal Operationalization)

目标	操作 1	操作 2
$C \Rightarrow \circ T$	**DomPre** $\neg T$ **DomPost** T **ReqTrig** C	**DomPre** T **DomPost** $\neg T$ **ReqTrig** $\neg C$
$C \Rightarrow \Diamond_{\leq d} T$	**DomPre** $\neg T$ **DomPost** T **ReqTrig** $\neg T \Diamond_{\leq d-1} C$	
$C \wedge T \Rightarrow \circ T$	**DomPre** T **DomPost** $\neg T$ **ReqTrig** $@C$	
$P \Rightarrow Q$	**DomPre** P **DomPost** $\neg P$ **ReqPost** Q	**DomPre** Q **DomPost** $\neg Q$ **ReqPost** $\neg P$
$C \Rightarrow \Box Q$	**DomPre** $\neg C$ **DomPost** C **ReqPost** Q	**DomPre** Q **DomPost** $\neg Q$ **ReqPre** $\blacksquare \neg C$
$C \Rightarrow \Box_{\leq d} Q$	**DomPre** $\neg C$ **DomPost** C **ReqPost** Q	**DomPre** Q **DomPost** $\neg Q$ **ReqPre** $\blacksquare_{\leq d-1} \neg C$
$\bullet C \Rightarrow Q \; W \; (Q \wedge R)$	**DomPre** $\neg Q$ **DomPost** Q **ReqTrig** C	**DomPre** Q **DomPost** $\neg Q$ **ReqTrig** $\neg C \; B \; (\neg C \wedge R)$
$@T \Rightarrow \bullet C$	**DomPre** $\neg T$ **DomPost** T **ReqPre** C	
$T \Rightarrow \bullet C$	**DomPre** $\neg T$ **DomPost** T **ReqPre** C	**DomPre** T **DomPost** $\neg T$ **ReqTrig** C

攻读博士期间发表主要论文

1. Zhang Yi, Zhang Yong, and Wang Weinong. Modeling and analyzing of workflow authorization management. Journal of Network and Systems Management, Kluwer Academic/Plenum Publishers, 2004 V12 (4), 2004, pp. 507-535. Accession number: 05068831188.
2. Zhang Yi, Zhang Yong, and Wang Weinong. A Complete Policy Lifecycle Model for Policy-based Security Management of Information Systems. 第四届中国信息和通信安全学术会议（CCICS'2005），中国西安，科学出版社，2005.5。
3. Zhang Yi, Xiaoli Liu, and Wang Weinong. Policy Lifecycle Model for Systems Management. IEEE IT Professional, IEEE Computer Society, March/April, 2005, pp. 50-54. Accession number: 05249152477.
4. Yi Zhang, Yong Zhang, Weinong Wang. Optimization of Firewall Filtering Rules by a Thorough Rewritten. 4th Latin American Network Operations and Management Symposium, Porto Alegre, Brazil, August 29-31, 2005.
5. Yi Zhang, Yong Zhang, Weinong Wang. Policy Engineering for Security Management of Organization Information Systems. 4th Latin American Network Operations and Management Symposium, Poster Paper, Porto Alegre, Brazil, August 29-31, 2005.
6. 张翼，张勇，汪为农。防火墙过滤规则的建模和全面优化，计算机工程与应用，2006，42（6）。
7. 张翼，张勇，汪为农。基于彩色 Petri 网的工作流权限管理的建模与分析计算机工程与设计，2006，11。
8. 张勇，张翼，汪为农。结合接收者访问控制的组密钥管理方案。计算机应用与软件。已录用。
9. Zhang Yong, Zhang Yi, Wang Wei Nong. A New Group Key Management Scheme Based on Keys Tree, XOR Operation and One-Way Function. Journal of SouthEast University (English). (Accepted).

攻读博士期间参加的科研工作

- 国家自然科学基金课题：网络入侵识别特征及分布式检测系统的研究；
- 国家信息安全办公室课题：基于防火墙日志的网络安全审计系统；
- 国家信息安全办公室课题：基于通用网络管理框架的网络安全管理技术。

致 谢

本人于 2001 年 9 月考入上海交通大学，师承我国教育科研网专家组十大专家之一，上海交通大学网络信息中心汪为农教授，攻读博士学位。本文从研究计划的制定、开题，到最后的撰写、成文，共历时三年多。在本文最终完成之际，特此向所有关心和支持我的人表示诚挚的谢意。

首先要深深地感谢我的导师，汪为农教授。没有汪老师建设性的建议、高屋建瓴的指导、热心的鼓励和全力的支持，也就不可能有本文的产生。感谢他对我的尊尊教诲、循循善诱，更要感谢他对我的理解与支持。汪老师严谨的治学态度、渊博的学识、教育事业的热情以及他特有的学者和长者风范，都对我产生了深刻的影响。

其次，我要感谢一直默默支持着我的家人。感谢我的父母，感谢他们从小对我的教育和培养；感谢我的妻子，感谢她的耐心与体贴；还要感谢我的儿子，为我的生活带来了无穷的乐趣。

另外，我还要感谢与我一同学习、一同科研攻关的师兄弟妹，特别是张勇、蒋俊杰、李承和程立等同学。与他们共同生活的日子让我永远难忘。祝愿他们前程似锦。

最后，我还要感谢上海交通大学网络信息中心的领导和老师们，感谢他们为我提供了良好的学习和科研环境。特别是顾一众主任、李家滨教授两位领导，以及周子豪、谢锐、瞿庆海、郭平、符兵、张云涛和茅维华等各位老师，感谢他们大家对我的关心和爱护。

张 翼

二 OO 六年三月

上海交通大学学位论文答辩决议书

申请者	张翼	所在学科（专业）	计算机应用技术
论文题目	信息和系统安全管理策略工程		
答辩日期	2006 年 6 月 6 日	地点	上海交通大学网络信息中心 浩然高科技大楼 4 楼

答辩委员会成员			
姓名	单位	职称	签名
高传善	复旦大学 计算机科学与工程系	教授	
白英彩	上海交通大学 计算机科学与工程系	教授	
马范援	上海交通大学 计算机科学与工程系	教授	
王能	华东师范大学 计算机科学技术系	教授	
陈涵生	华东计算所	教授	

评语和决议：

策略在系统管理中扮演着重要的角色。然而在一个复杂的信息系统中， 管理策略本身就构成了一个复杂的子系统，并形成于一个复杂而又完整的过程。因此管理策略的研究成为成功的系统管理的关键，也是目前该领域中的薄弱环节之一。论文的选题针对该研究领域的薄弱环节，给出了具有建设性的研究成果，对于信息技术在国民经济中的具体应用和推广具有重要的理论意义和实用价值。论文的主要工作和成果如下；（1）系统地研究了管理策略的生命周期，提出了策略工程和策略生命周期的概念，并给出了一个合理的策略生命周期模型；（2）借鉴已有的需求分析技术，提出了基于目标的企业信息和系统安全管理需求分析框架模型，给出了建立目标模型的步骤和方法以及定义策略的规则，并讨论了如何建立反目标模型和根据反目标模型确定管理策略的方法；（3）对策略的分析和验证，提出了一些新的建模和分析方法，具有较高的实用价值。论文结合了信息技术和管理学两个学科的研究成果，是一种有益的探索，成果有创新性。论文立意正确，内容充实，行文流畅，条理清晰，反映作者具有坚实宽广的理论基础、深入系统的专业知识，独立科研工作能力强。答辩过程中叙述清晰，回答问题正确。经答辩委员会无记名投票，一致同意通过张翼同学的博士论文答辩，建议授予工学博士学位。

表决结果：

答辩委员会主席 （签名）

年 月 日