# 第一章 绪论

电信网网络管理是随着电信行业和计算机产业的发展而成长起来的。在步进、纵横交换制为主的通信年代，电信网还不需要什么网络管理和服务响应，设备维护是当时最实际的需要。后来程控交换机大量步入电信网内，电信网络的规模在迅速增长，当网络的用户在不断膨胀，设备维护的人力成本和效率已经大大影响到电信运营商的运行维护成本时，电信网络的运营商意识到，他们需要使用一种新的、高效的、自动的、智能的管理工具来保证网络的正常运行[1]。

电信网网络管理技术正是在这种需求下应运而生的。当然，计算机产业的飞速发展无疑为以高效、自动、智能为目标的电信网网络管理技术提供了根本的保证。基于计算机平台和各种软件系统的电信网网管系统实现了对电信网络高效、自动、智能的监测与管理，满足了电信运营者降低网络运营成本、提供更多业务、提高服务质量的需求。

根据这些需求，国际电信联盟电信标准化部门(ITU-T)提出了电信管理网络 (TMN)规范，作为对移动通信网络管理的基础。TMN 作为一个单独的网络，它与移动网络在逻辑上是分开的，它通过一些标准的功能接口与移动网络相连接，并通过这些接口与移动网络进行通信，对网络中的设备进行管理和监测。同时，在物理上 TMN 与移动网络又可以是集成在一起的，TMN 可以通过移动网络提供的服务进行通信[2]。TMN 是基于 OSI 标准制定的，它采用了 OSI 管理中的面向对象技术对网络资源进行描述和管理。它的目标是为不同的通信设备提供标准的模型和接口。从管理功能上，它定义了配置管理、故障管理、性能管理、安全管理和计费管理等五个网络管理功能。每个功能都侧重于某一方面的管理，但它们并不是各自独立的，在一定条件下这五个功能可以相互转化。

移动通信网是电信网的一个重要组成部分，近年来，随着移动通信技术的发展，移动通信网络也在不断的发展壮大，移动网络的覆盖范围和复杂程度都有很大的扩展和提高。因此，对移动网络管理系统的要求也随之越来越高，仅仅依靠各个设备自带的管理工具进行分别管理已经不能满足网络发展的需要。现代化的移动网络需要一个覆盖面广、自动化程度高并可以灵活配置和扩展的标准的移动网络管理系统。同时随着 3G 的发展，3G 移动网管应运而生。3G 移动网管和 2G 移动网管在需求和系统实现上是有区别和联系的。其相同点如具有管理同一系统中不同厂家设备的能力；能够在 UMTS 网元与网管之间以及网管系统自身之间通过标准接口进行通讯；提供灵活配置能力，允许新业务迅速开展等。其不同点有如 3G 网管与 2G 网管基于的参考模型不同；3G 网管具有更加丰富的业务管理能力；3G 网管与 2G 网管对外接口不同，2G 主要是 Q3 接口，3G 比较认可 CORBA 接口。

## 1.1 论文的选题背景及意义

随着通信技术的高速发展，第三代移动通信系统(3G)技术的成熟和商用，移动网络规模的不断扩大，网络结构的不断地变化，网络复杂性的日益提高，对网络管理和维护变得越来越重要。为了确保网络处于高效的运营状态，及时的了解网络运营的相关数据，提高网络的运行效率和缩短故障的修复时间，迫切需要开发先进的管理系统与之相配套。在这种大环境的要求下，网络管理系统已成为移动通信网络建设中的热点、焦点和难点问题。

本文通过介绍网络管理系统的相关背景知识，对移动网管体系结构及关键技术进行了深入的研究，结合 3G 网络的特点，给出了一个基于 TMN 分层思想的网管体系结构，并结合软件管理模块的设计重点讨论了网元管理系统的实现。此论文可以为 3G 移动网管以及其他类型的网络管理研究提供一些参考价值。

## 1.2 网管系统的发展现状和发展方向

在九十年代，国内的一些高校和研究单位开始对移动网络管理技术进行研究，并借鉴了大量国外的先进的网络管理技术，通过资料来看，国外的网络管理技术比我国要成熟得多，尤其是在数据库、分布式处理、中间件交换平台等上的应用技术比我国先进了很多。

近几年来，国内移动运营商的网络规模不断扩大，也逐渐加大了对网管技术的研究，目前在 3G 网管的接口标准化方面的研究已得到许多国际标准化组织的承认。

## 1.3 本论文的主要工作

根据国内外在 3G 网络管理方面的研究、开发现状，以及最新发展动态，特别是网元管理层的研究进展，我们拟在系统实现架构、网元接入、软件管理等几个方面对 3G 移动网络管理系统的设计与实现展开深入研究。具体而言，论文将在以下几个方面展开研究。

**(1) 3G 移动网管系统实现架构以及网元管理系统的内部架构**

结合 TMN 的分层实现思想，考虑当前 3G 网络发展的现状，给出了一个从网络管理层到网元层的管理模型，讨论了不同层次之间的网管系统的接口问题；结合 CORBA 技术，以及其他的一些中间件技术，设计出一套分布式、可扩充、跨平台的高效的网元管理系统。

**(2) 网元的动态接入**

网元的接入是网元管理系统中的一个关键问题，传统的网元接入费时费力，而且不利于系统的升级改造。我们尝试使用 Manager/Agent 结构，结合 XML 技术，给出了一个网元动态接入方案。

**(3) 软件管理模块的实现**

软件管理是网络管理系统的重要组成部分，我们主要利用此模块对通信网络中各个网元节点上的一些重要文件进行统一的管理维护。主要的操作包括各种文件单元在 GUI、网管服务器和网元之间的传输，以及网元管理系统（服务器或 GUI）对这些传送内容的处理。

## 1.4 论文结构

本文各主要章节内容安排如下：

第一章作为整个论文的绪论，介绍网络管理系统的背景知识和国内外网管技术的发展现状，描述了论文的选题背景和意义，给出了课题的研究方案，在叙述了研究内容后，简要地介绍了论文结构。

第二章介绍了当前流行的网络管理技术及其发展方向。

第三章详细介绍了网管系统的模型基础——TMN 网管模型，讨论了 TMN 的网络管理功能及其体系结构，在本章的最后还介绍了 Q3 通信接口。

第四章对 3G 移动网管做了规划，根据 3G 网络的特点，提出了 3G 网络管理系统应具备的管理功能。并讨论了具体的实现方案，在本章的最后，讨论了具体实现中的关键技术：中间件技术，CORBA 技术，XML 技术和网元动态接入技术，并给出了具体系统的实现框图。

第五章针对 3G 网管中的软件管理功能进行了分析，阐述了软件管理的具体业务功能和需求，并对软件管理中涉及到的几个主要业务流程进行了分析。

第六章讨论了软件管理模块的具体实现。软件管理系统的实现主要由三个部分构成：服务子系统，适配子系统，GUI 子系统，在本章中，分别对每一个模块的设计和实现进行了介绍。

第七章总结了本论文的研究成果与创新点，并提出了进一步的研究方向。

# 第二章 当前流行的网管实现技术及发展趋势

随着网管技术得不断进步，网络管理系统在自动化和智能化方面有了很大提升，不断地有新技术和新功能推出。本章将从网络协议、网管技术等方面简要介绍一下网络管理系统采用的技术及发展趋势。

## 2.1 网管技术使用的主要协议

虽然各个网管系统彼此提供的性能不同，但是基本上都采用了 SNMP、DMI、WMI、TCP/IP、SPX/IPX、SNA、DECNET、SAN 等协议。SNMP 是由一系列协议组和规范组成的，它们提供了一种从网络上的设备中收集网络管理信息的方法。另外，有些网络管理软件采用了 CMIP 协议（一种较 SNMP 更为详细的网络管理协议，但由于其自身的一些缺陷，并未被广泛使用）[3]。

## 2.2 网管软件的主要技术及方向

随着网络管理需求的不断增加，越来越多的网络管理技术被开发和使用，下面简要介绍网络管理领域相关的一些最新技术及其应用。

### 2.2.1 Portal 技术

Portal 是一个基于浏览器的、建立和开发企业信息门户的软件环境，具有很强的可扩展性、兼容性和综合性。它提供了对分布式软件服务和信息资源的安全、可管理的框架。便于使用的 Portal 界面为每个用户提供了他所需要的信息和 Web 内容，同时也保证了每个用户只能访问他所能访问的信息资源和应用逻辑。

### 2.2.2 RMON 技术

网络管理技术的一个新的趋势是使用 RMON（远程网络监控）。RMON 的目标是为了扩展 SNMP 的 MIB－II（管理信息库），使 SNMP 更为有效、更为积极主动地监控远程设备。

### 2.2.3 基于 Web 的网络管理技术

由于 Web 有独立的平台，且易于控制和使用，因而常被用来实现可视化的显示。

### 2.2.4 XML 技术

采用 XML 技术，系统提供了标准的信息源，可以与企业内部的其它专业系统或外部系统进行数据交互。

### 2.2.5 CORBA 技术

CORBA 是 OMG(Object Management Group)为解决不同软硬件产品之间互操作而提出的一种解决方案。简单地说，CORBA 是一个面向对象的分布式计算平台，它允许不同的程序之间可以透明地进行互操作，而不用关心对方位于何地、由谁来设计、运行于何种软硬件平台以及用何种语言实现等，从而使不同的网络管理模式能够结合在一起。

## 2.3 网管系统的发展趋势

总体来说，网管软件的发展趋势将体现在以下几个方面[4]：

**系统平台化**

网管综合化一直是用户追求的目标。系统平台化因其适配灵活、构筑方便且扩展性好，慢慢成为系统的发展方向。未来的网管系统是能够管理多个不同的网络系统的，如管理者在管理平台上，能够管理 PSTN 网，移动通信网以及 IP 网络等等。

**管理更集中**

网管系统从开始就是体现了集中的思想。首先是集中维护，然后有了集中监控、集中管理，在一个管理点上能够实现对分布在通信网中各个网元的管理。

**处理更分布**

与建设规模更集中相反，系统的处理方式将更分布。一方面由于更大规模的集中导致系统处理负荷急剧增加，从负载平衡和健壮性的角度考虑，分布式的处理都是最佳的解决办法。另一方面合理的分布方式，有效地提供了系统的扩展能力，也为大规模集中提供解决途径。

**与资源结合紧密**

网管(NMS)和资源管理(RMS)是 OSS 的重要组成部分。资源管理偏重于资源数据的静态管理和调度，而网管系统从某种程度上也可以认为是资源管理的一种监测、控制和实施工具。网管系统只有与被管网络的资源数据有机结合在一起，才能够真正做到动态、智能化的管理和维护。

**更多面向业务**

电信行业的市场化直接影响到了网管的发展变化，这种影响是方向性的。随着业务类型的不断更新，对于工单的处理能力要求必然日益提高，网管系统必然在体系上要保证对工单的控制、管理和实施及时有效。同时，对于设备和网络的控制将越来越多地与工作流程和预案管理相结合，网管的智能化程度将得到更大的提升，网管的业务快速实施能力必然得到极大的加强。

总的来说，网管系统软件的平台化，管理集中化，更多面向业务等是目前及以后网管技术的一种趋势，各网络管理系统软件提供商开发和应用的重点也正随着做出调整。对于网络系统，网管将开始扮演越来越重要的角色。

# 第三章 TMN 网管模型及其体系结构介绍

TMN（TelecommunicationsManagementNetwork）是 ITU－T 提出来的关于网络管理系统化的解决方案，是网管领域的热门话题。TMN 的基本概念是提供了一个有组织的网络结构，以取得各种类型的操作系统之间以及操作系统与电信设备之间的互联。它是一个完整的独立的管理网络，是各种不同应用的管理系统按照 TMN 标准接口互联而成的网络。M.3010 建议给出了 TMN 的原则，规范性地描述了一个电信管理网的层次结构、功能结构、逻辑结构和以 CMIP / GDMO 为核心的接口规程与信息模型[5]。

作为网管系统的模型基础，本章将向读者详细介绍 TMN 模型。

## 3.1 TMN 定义的网管管理功能

TMN 提供了一种有组织的体系结构，实现各种不同类型的运营系统（Operation System）与电信网络设备之间的互联，通过包括协议和消息在内的标准化接口来交换各种管理信息。

TMN 是一种标准，是 OSI 网络管理方案在电信领域的应用和发展。OSI 对系统的管理提出了概念模型，而 TMN 则突出了网络管理这一概念，通过业务对网络进行管理，实现了管理网与被管的通信网的业务分离，营造了网络管理可持续发展的环境。

TMN 是一个指导思想，它定义了可遵循的、基于 Q3/CMIP 的标准接口，以及 Q3 适配器，以便接入系统的人机管理终端和非 TMN 标准的设备，实现了后向兼容；同时，它又可以通过关口的方式，与完全面向对象的网络管理平台，例如 OMG（开发管理组）制定的通用对象请求代理结构（CORBA），进行综合，以支持前向兼容。另一方面，TMN 提出的逻辑分层体系结构，不仅针对电信网络的管理，而且也是电信运营公司用以构造业务运营支撑系统以及构建企业信息支持系统的指导原则。近年来，在 TMN 演进的过程中，由网络管理论坛（NMF）所知道的电信运营图（Telecom Operation Map）就是 TMN 指导思想的发展[6]。

TMN 是一个框架，它定义了电信网络管理系统的功能体系结构、逻辑分层体系结构、物理体系结构和信息体系结构。

TMN 的目标是支持电信运营部门的各种管理需要，对电信网及其业务进行规划，提供安装、维护、运营和管理。TMN 能使运营部门对网络事件作出最迅速的反应，优化管理信息流，进行地域分布控制，提高对业务的支撑力度，以及改善与用户的交互渠道，逐步实现以用户为中心的管理模式。

OSI 中定义了网络管理中的功能模型组件，它强调了面向用户的应用，如图 3.1 所示。该功能模型由五个子模型构成：配置管理、故障管理、性能管理、安全管理和计费管理[7]。

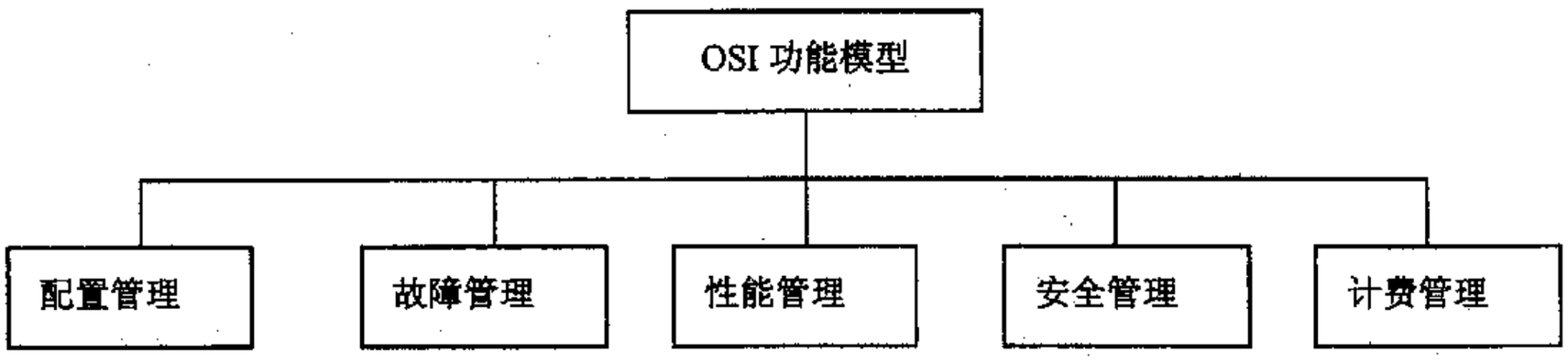



图 3.1 网络管理功能模型

### 3.1.1 配置管理

网络上的配置是网络上各种工作设备、备份设备、辅助设备之间关系的状态。为了保证网络经济、可靠、高效和安全地运行，需要对网络上的配置进行调整。对网络上配置进行调

整的管理活动，就是配置管理[8]。从网络的运行角度看，配置管理分为两个阶段：网络初次运行的初始配置管理（通常称为网络指配，也称为网络预配置）和网络正常运行时的工作配置管理（通常称为配置控制）。

配置管理具体包括如下功能：

(1) 配置信息的自动获取：在一个大型网络中，需要管理的设备是比较多的，如果每个设备的配置信息都完全依靠管理人员的手工输入，工作量是相当大的，而且还存在出错的可能性。对于不熟悉网络结构的人员来说，这项工作甚至无法完成，因此，一个先进的网络管理系统应该具有配置信息自动获取功能。即使在管理人员不是很熟悉网络结构和配置状况的情况下，也能通过有关的技术手段来完成对网络的配置和管理。在网络设备的配置信息中，根据获取手段大致可以分为三类：一类是网络管理协议标准的 MIB 中定义的配置信息(包括 SNMP 和 CMIP 协议)；二类是不在网络管理协议标准中有定义，但是对设备运行比较重要的配置信息；三类就是用于管理的一些辅助信息。

(2) 自动配置、自动备份及相关技术：配置信息自动获取功能相当于从网络设备中“读”信息，相应的，在网络管理应用中还有大量“写”信息的需求。同样根据设置手段对网络配置信息进行分类：一类是可以通过网络管理协议标准中定义的方法(如 SNMP 中的 Set 服务)进行设置的配置信息；二类是可以通过自动登录到设备进行配置的信息；三类就是需要修改的管理性配置信息。

(3) 配置一致性检查：在一个大型网络中，由于网络设备众多，而且由于管理的原因，这些设备很可能不是由同一个管理人员进行配置的。实际上即使是同一个管理员对设备进行的配置，也会由于各种原因导致配置一致性问题。因此，对整个网络的配置情况进行一致性检查是必需的。在网络的配置中，对电信网络正常运行影响最大的主要是局向信息配置，因此，要进行一致性检查的也主要是这类信息。

(4) 用户操作记录功能：配置系统的安全性是整个网络管理系统安全的核心，因此，必须对用户进行的每一配置操作进行记录。在配置管理中，需要对用户操作进行记录，并保存下来。管理人员可以随时查看特定用户在特定时间内进行的特定配置操作。

### 3.1.2 故障管理

故障管理是对网络发生异常情况——故障时所采取的一系列管理活动。这一系列活动包括故障管理有关的管理参数的确定、故障指标管理、故障监视/测试、故障定位和故障恢复[9]。

(1) 故障监测：主动探测或被动接收网络上的各种事件信息，并识别出其中与网络和系统故障相关的内容，对其中的关键部分保持跟踪，生成网络故障事件记录。

(2) 故障报警：接收故障监测模块传来的报警信息，根据报警策略驱动不同的报警程序，以报警窗口/告警铃声(通知一线网络管理人员)、电子邮件或短消息(通知决策管理人员)发出网络严重故障警报。

(3) 故障信息管理：依靠对事件记录的分析，定义网络故障并生成故障卡片，记录排除故障的步骤和与故障相关的值班员日志，构造排错行动记录，将事件-故障-日志构成逻辑上相互关联的整体，以反映故障产生、变化、消除的整个过程的各个方面。

(4) 排错支持工具：向管理人员提供一系列的实时检测工具，对被管设备的状况进行测试并记录下测试结果以供技术人员分析和排错；根据已有的排错经验和管理员对故障状态的描述给出对排错行动的提示。

(5) 检索/分析故障信息：浏览并且以关键字检索查询故障管理系统中所有的数据库记录，定期收集故障记录数据，在此基础上给出被管网络系统、被管线路设备的可靠性参数。

### 3.1.3 性能管理

网络管理就是对网络的运行状态进行管理，当网络产生故障时，由故障管理进行管理。

当网络没有产生故障，或者没有产生能让故障管理进行管理的故障时，由于各种原因导致网络质量或服务质量下降，就要使用性能管理。主要包括性能管理参数管理、性能指标管理、性能监控、性能分析、性能控制等[10]。

(1) 性能监控：由用户定义被管对象及其属性。被管对象类型包括链路、单板、CPU 等；被管对象属性包括流量、延迟、丢包率、CPU 利用率、温度、内存余量。对于每个被管对象，定时采集性能数据，自动生成性能报告。

(2) 阈值控制：可对每一个被管对象的每一个指标设置阈值，对于特定被管对象的特定指标，可以针对不同的告警级别进行阈值设置。可通过设置阈值检查开关控制阈值检查和告警，提供相应的阈值管理和溢出告警机制。

(3) 性能分析：对历史数据进行分析，统计和整理，计算性能指标，对性能状况做出判断，为网络规划和网络优化提供参考。

(4) 可视化性能报告：对数据进行扫描和处理，生成性能趋势曲线，以直观图形反映性能分析的结果。

(5) 实时性能监控：提供了一系列实时数据采集；分析和可视化工具，用以对流量、负载、丢包、温度、内存、延迟等网络设备和链路的性能指标进行实时检测，可任意设置数据采集间隔。

(6) 网络对象性能查询：可通过列表或按关键字检索被管网络对象及其属性的性能记录。

### 3.1.4 安全管理

网络管理过程中，存储和传输的管理和控制信息对网络的运行和管理至关重要，一旦泄密、被篡改和伪造，将给网络造成灾难性的破坏。安全管理的功能分为两部分，首先是网络管理本身的安全，其次是被管网络对象的安全。网络管理本身的安全由以下机制来保证：

(1) 管理员身份认证，采用基于公开密钥的证书认证机制；为提高系统效率，对于信任域内(如局域网)的用户，可以使用简单口令认证。

(2) 管理信息存储和传输的加密与完整性，Web 浏览器和网络管理服务器之间采用安全套接字层(SSL)传输协议，对管理信息加密传输并保证其完整性；内部存储机密信息，如登录口令等，也是经过加密的。

(3) 网络管理用户分组管理与访问控制，网络管理系统的用户(即管理员)按任务的不同分成若干用户组，不同的用户组中有不同的权限范围，对用户的操作由访问控制检查，保证用户不能越权使用网络管理系统。

(4) 系统日志分析，记录用户所有的操作，使系统的操作和对网络对象的修改有据可查，同时也有助于故障的跟踪与恢复。

网络对象的安全管理有以下功能：

(1) 网络资源的访问控制，通过管理路由器的访问控制链表，完成防火墙的管理功能，即从网络层(IP)和传输层(TCP)控制对网络资源的访问，保护网络内部的设备和应用服务，防止外来的攻击。

(2) 告警事件分析，接收网络对象所发出的告警事件，分析与安全相关的信息(如路由器登录信息、SNMP 认证失败信息)，实时地向管理员告警，并提供历史安全事件的检索与分析机制，及时地发现正在进行的攻击或可疑的攻击迹象。

(3) 主机系统的安全漏洞检测，实时的监测主机系统的重要服务(如 WWW，DNS 等)的状态，提供安全监测工具，以搜索系统可能存在的安全漏洞或安全隐患，并给出弥补的措施。

总之，网络管理通过网关(即边界路由器)控制外来用户对网络资源的访问，以防止外来的攻击；通过告警事件的分析处理，以发现正在进行的可能的攻击；通过安全漏洞检查来发现存在的安全隐患，以防患于未然。

### 3.1.5 帐务管理

计费管理的主要范围是计费及其有关的财务管理。

(1) 计费数据采集：计费数据采集是整个计费系统的基础，但计费数据采集往往受到采集设备硬件与软件的制约，而且也与进行计费的网络资源有关。

(2) 数据管理与数据维护：计费管理人工交互性很强，虽然有很多数据维护系统自动完成，但仍然需要人为管理，包括交纳费用的输入、联网单位信息维护，以及账单样式决定等。

(3) 计费政策制定：由于计费政策经常灵活变化，因此实现用户自由制定输入计费政策尤其重要。这样需要一个制定计费政策的友好人机界面和完善的实现计费政策的数据模型。

(4) 政策比较与决策支持：计费管理应该提供多套计费政策的数据比较，为政策制订提供决策依据。

(5) 数据分析与费用计算：利用采集的网络资源使用数据，联网用户的详细信息以及计费政策计算网络用户资源的使用情况，并计算出应交纳的费用。

(6) 数据查询：提供给每个网络用户关于自身使用网络资源情况的详细信息，网络用户根据这些信息可以计算、核对自己的收费情况。

## 3.2 TMN 的体系结构

在 M.3010 规范中定义了 TMN 的体系结构，包括不同的 TMN 实体以及实体间的通信。体系结构可以从 4 个方面进行描述：功能体系结构、物理体系结构、逻辑分层结构和信息体系结构。

### 3.2.1 TMN 的功能体系结构

功能体系结构有以下几个实体：OSF（运营系统功能）、QAF（协议适配功能）、MF（中介功能）、NEF（网元功能）和 WSF（工作站功能）。每一个实体代表一个功能块，不同功能块之间的通信是标准化的（至少是将被标准化的），这些标准化的通信点称作 TMN 参考点（TMN Reference Point），它也是功能块分隔的概念性边界[11]。图 3.2 是 TMN 的功能体系结构的示意图。

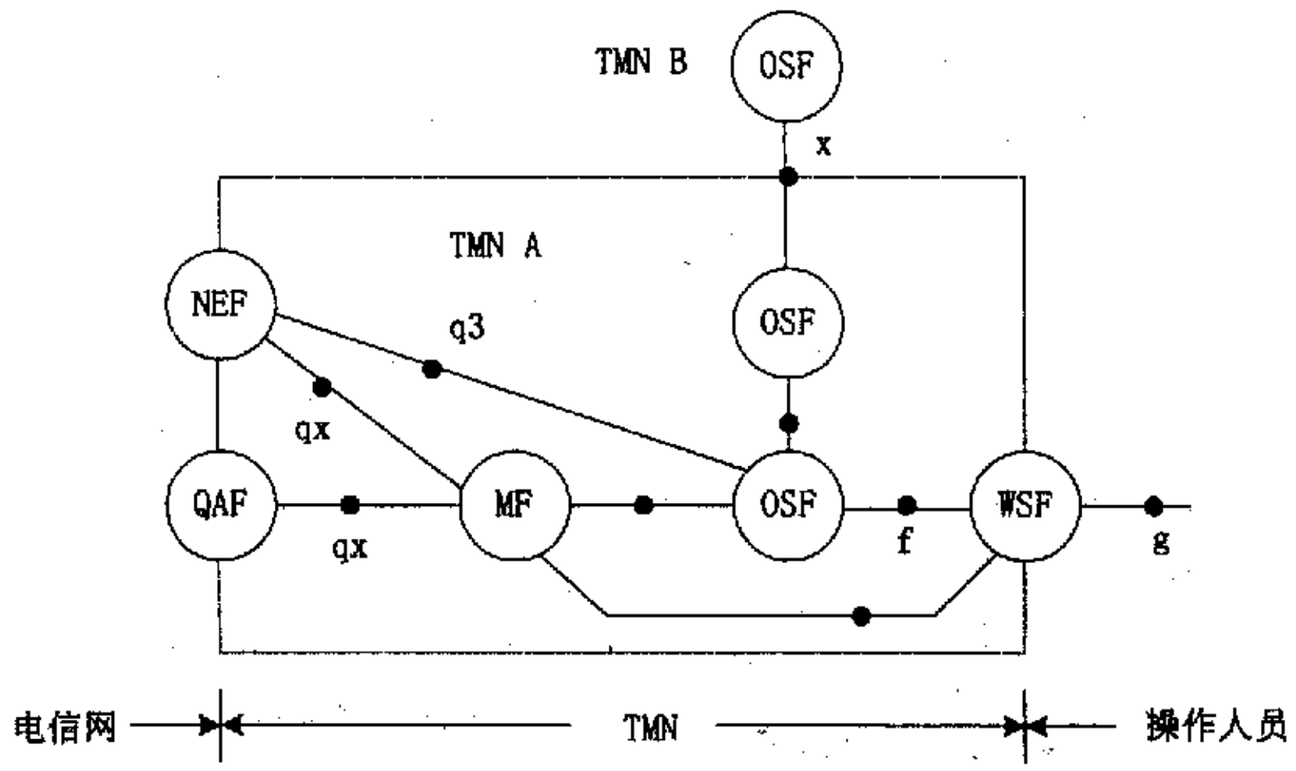



图 3.2 TMN 的功能体系结构

从图 3.2 可以清晰地看到 TMN 与电信网的边界，与操作员的边界以及与其他网管系统的边界。在 TMN 与电信网的边界上有两个功能块：NEF 和 QAF。NEF 代表网元功能，它和 MF 之间的参考点是 qx，和 OSF 之间的参考点是 q3。QAF 代表 Q 接口适配功能，它用于接入非标准 TMN 的传统系统，它与 MF 之间的参考点是 qx。MF 代表中介功能，它和

NEF 和 QAF 之间的参考点是 qx，和 OSF 之间的参考点是 q3，和 WSF 之间的参考点是 f。OSF 代表运营系统功能，它是处理管理信息的核心，在它和 MF，NEF 和 OSF 之间是 q3 参考点，和其他 TMN 系统的 OSF 之间是 x 参考点，和 WSF 之间是 f 参考点。WSF 代表工作站功能，它位于 TMN 同操作员的边界，负责 TMN 与人之间的交互，包括各种输入输出和界面，它和 OSF 和 MF 之间是 f 参考点，它和操作员之间是 g 参考点。

### 3.2.2 TMN 的物理体系结构

TMN 物理体系结构由物理实体及其相互关系（见图 3.3）。其中每个物理块是一个或多个功能块的实现，每个物理块的命名是依据其必需实现的功能块来决定的。例如，OS 必须实现 OSF 的功能，除此之外，它还可以实现 MF 或 WSF 等功能块，但不管是什么 OS，OSF 功能是它必须实现的功能，其他功能块都是可选的。物理块还有 DCN（数据通信网），NE（网元），MD（中介设备），WS（工作站）和 QA（Q 适配器），它们都有自己相应的必选功能块，用物理块的名字后面加上 F 来表示。对应于功能体系结构中的参考点，不同的物理块之间的通信被称为接口，已知的接口有 Q3 接口、Qx 接口、X 接口和 F 接口，每一个接口都是相应参考点的实现[12]。假如一个电信网络设备具有向上提供 Q3 接口或 Qx 接口的能力，那么这个网络设备就是一个网元(NE)，因为它具备了 NE 所必须具备的功能块，这个网元还可能实现了其他的功能块，如 MF 功能块。

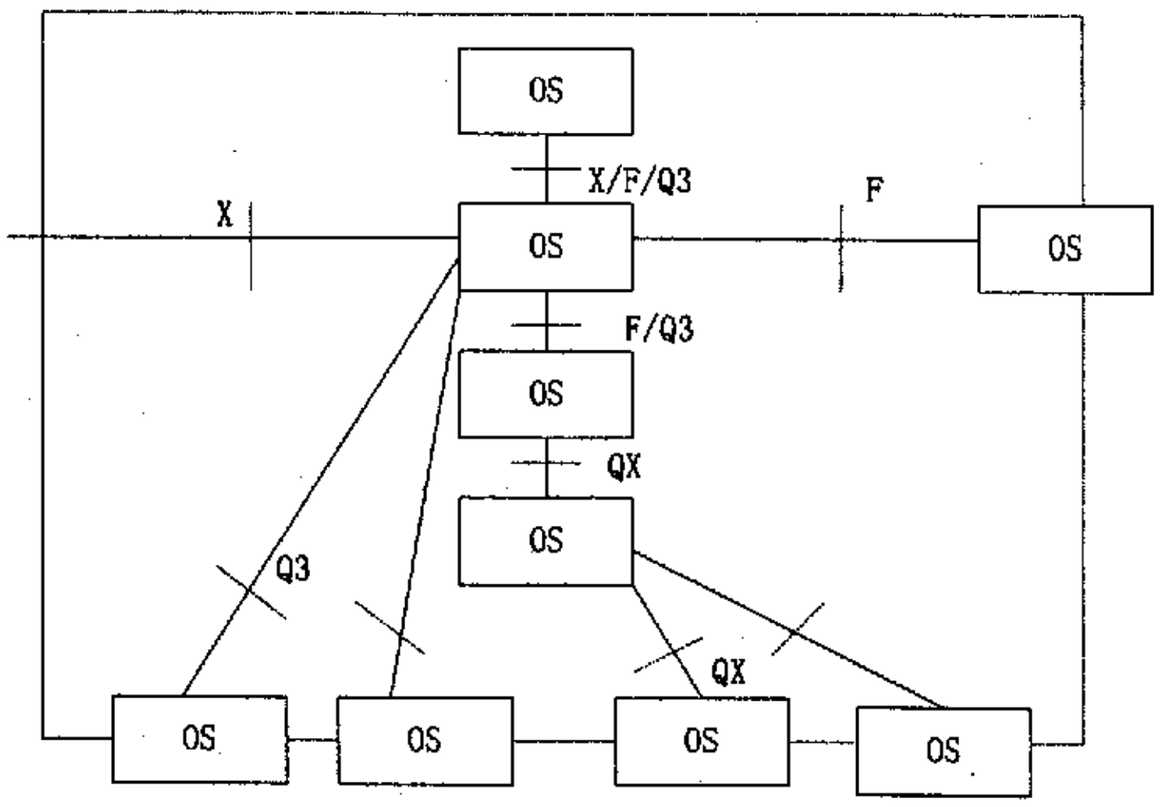



图 3.3 物理体系结构

### 3.2.3 TMN 的逻辑分层体系结构

对 TMN 的功能体系结构的一个扩展就是它的逻辑分层体系结构，它将 OS 的管理功能分为如下 4 个层次[13]。

(1) 网元管理层

它管理的若干个 NE 的组合，例如一个子网。它关心的往往是与具体网元相关的状态和操作等，提供最基本的非集中式的底层管理功能，如性能数据的收集、筛选和分析、告警收集及协议，其作用是为高层提供获取系统资源的手段。因此网元管理常常与某一特定的厂商相关。

(2) 网络管理网

它管理的是整个网络，从范围上讲，它涉及整个网络中所有的网元以及网元之间的连接，例如交换设备和传输设备；从内容上讲，它关心的是独立于具体厂商的带有普遍意义的指标和信息。它还提供对网络中各种业务的支持。

(3) 业务管理层

它管理的是对用户提供的各种业务，例如处理各种业务定单、投诉、孤枕单、计费和

QoS 测量等。同网元管理和网络管理相比较，它对网络上具体应用技术的依赖较小。

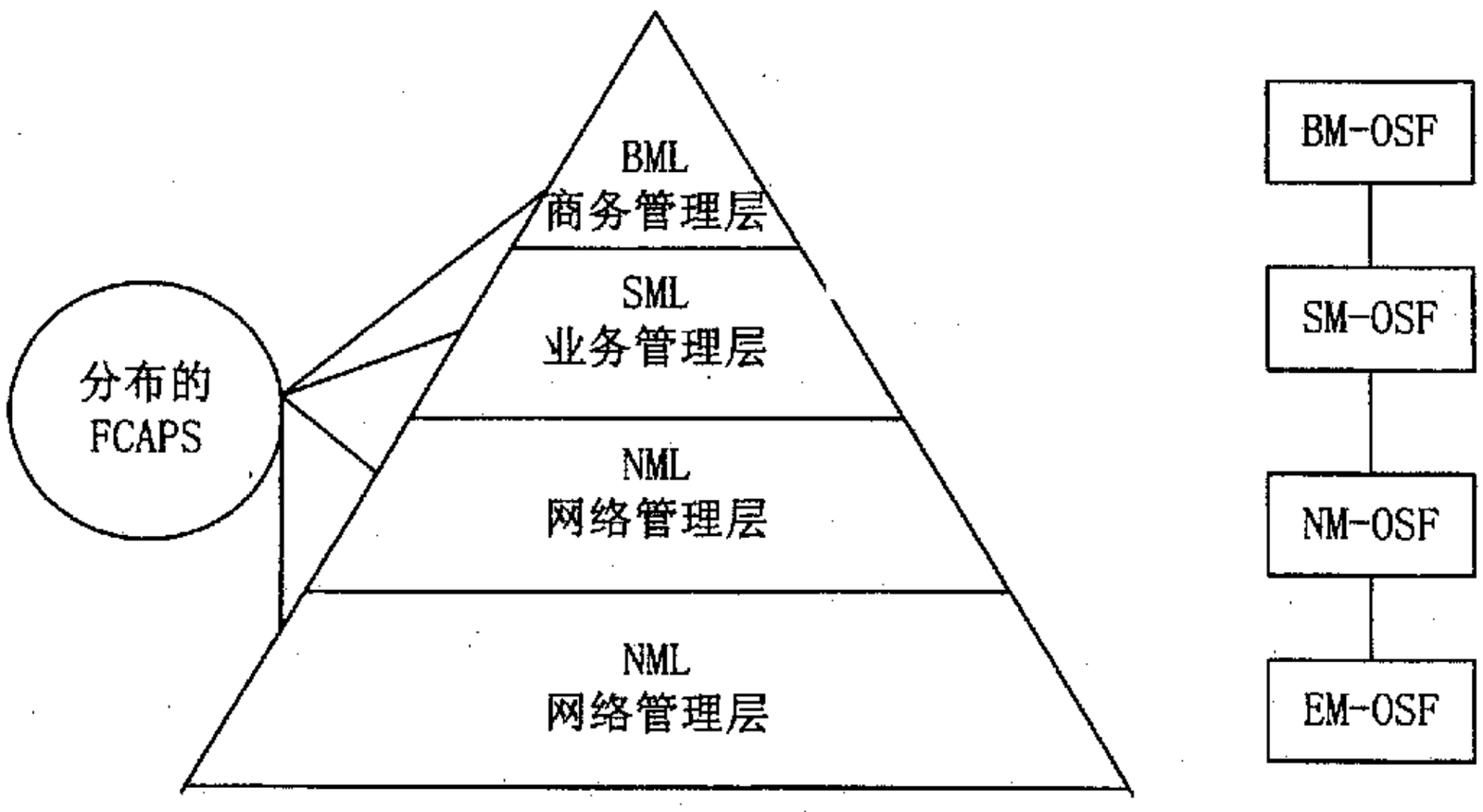
（4）商务管理层

这是网络管理的最高层次，它管理的往往是一个电信运营公司的决策者所关心的事务。它应该支持电信企业对资金投入的决策过程，包括需重点提供的业务，主要的市场范畴及资金效率；它也应该支持企业发展目标的确定过程，如成本、利润、增长率等，甚至包括人力资源协调等功能，是企业的管理信息系统(MIS)。“Business Management”也可以翻译为事务管理，但商务这个词更能体现与业务的联系和区别。

从网元管理层到商务管理层，所管理的由微观到宏观由操作层面到规划层面，相应的，使用者也在发生变化。在分层的概念中，每一个逻辑层面都向上一层提供信息，上一层拥有比下一层更高的信息抽象。

一个具体的网管系统可以全部实现这 4 个层次的管理，但目前的网管产品大多集中在实现前 3 个管理层次上。从实现的角度讲，层次越高，实现就越复杂，只实现其中的一个网元层的系统称为网元管理系统。本课题实现的综合故障管理系统正是集成与网元管理系统中。

TMN 的逻辑分层结构表示了各种管理活动的分层情况。每一层都由相应的功能实体来实现它的功能，分别取名为 BM-OSF，SM-OSF，NM-OSF 和 EM-OSF，构成如图 3.4 所示的管理层次模型。需要强调的示，各层的 OSF 都包括故障管理（FM）、配置管理（CM）、帐务管理（AM），性能管理（PM）和安全管理（SM）等功能，缩写为 FCAPS[14]。但是，各层 FCAPS 所完成的具体功能却各不相同。例如，性能管理可细分为性能质量保证、性能监视、性能管理控制和性能分析等 4 个子功能。我们仅以性能分析子功能作为例子来说明功能的分布：在网元管理层需进行针对网元的话务量容量分析；在网络管理层进行全网话务量分析；在业务管理层则需要综合分析用户的话务量性能；最后，在最高的商务管理层进行话务量预测并作出性能改善的建议。




(a)逻辑层　　(b)管理层次模型

图 3.4 逻辑分层体系结构

### 3.2.4 TMN 的信息体系结构

TMN 的信息体系结构是建立在面向对象的概念上的。它定义了接口两侧的实体如何进行信息交换。首先，定义了信息交换过程中的两个角色：管理者角色和代理角色（图 3.5）。扮演管理者角色的实体在信息交换中向代理发出指示，获得响应，并接受来自代理的通知；扮演代理角色的实体执行管理者的指示，发回响应，并在必要时向管理者发出通知，使信息得以在被管理的电信网络设备与管理用的计算机网络设备之间进行交换。在信息交互的过程

中，网络资源被抽象为一个个被管理对象（Managed Object），被管理对象可能是网络资源某些特性集的抽象，也可能是网络资源之间关系的抽象，还有可能是管理系统自身的软硬件组件的抽象。代理接到管理者的指示后，对网络资源执行操作，代理本身不是被管理的网络资源，它不是网络操作的终极目标，但是它的作用却是不容忽视的，经过它，管理者看到的不再是具体的网络资源，而是整齐划一的管理对象。

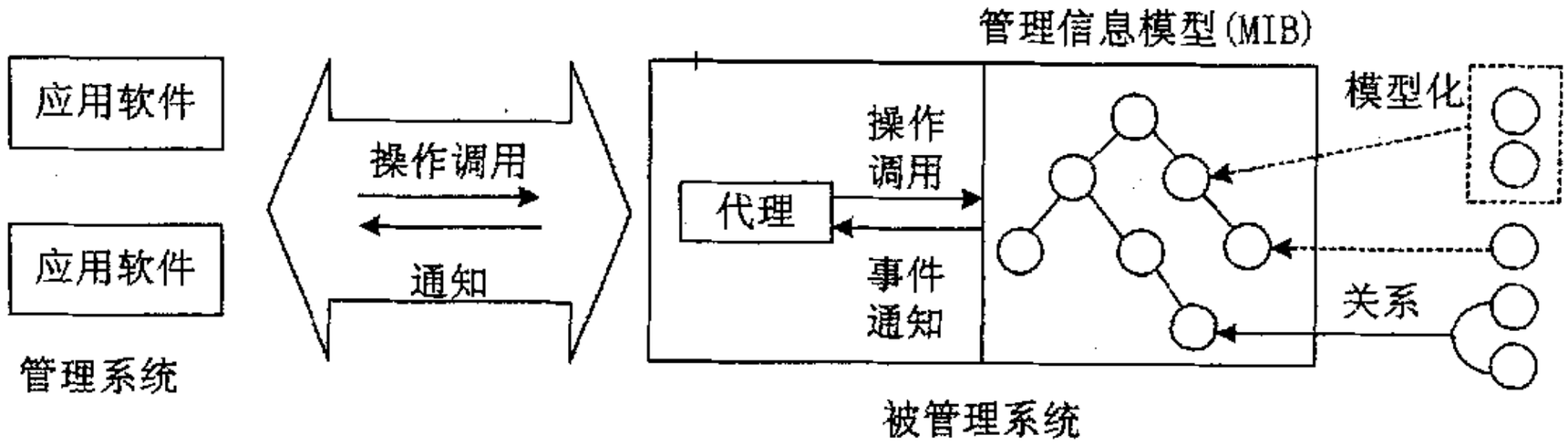



图 3.5 信息体系结构

需要指出的是，被管理对象与实际的被管理网络资源之间并不总是一一对应。一个对象可以代表多种资源，或多个对象代表一种资源，也可以用一个对象来表示由各个子实体（即对象的子类）所组成的一种复杂的资源。

由于 TMN 功能块中描述的管理活动通过管理者和代理的结构来实现，因此，可以采用面向对象的技术(OOT)来开发应用软件。OOT 包括了面向对象的分析(OOA)、设计(OOD)、编程（OOP）以及数据库技术(OODB)。从这个角度出发，软件系统就成为包括数据结构和行为两个方面的对象的集合，实现了对象的外部性质与其内部实现的分离，不仅增强了对象的重用性，而且也减弱了应用程序的依赖性。

管理者和代理之间要进行信息交互，必须对这些管理对象的定义达成共识。这就引出了 TMN 中另一个重要概念——信息模型。针对不同的被管理网络，需要定义不同的信息模型，如用于 SDH 网络管理的信息模型、用于接入网管理的信息模型，以及用于移动通信网管理的信息模型等等。

## 3.3 Q3 接口

Q3 接口不同于一般意义上的通信接口，它由两个部分组成：跨越 OSI 七层参考模型的协议栈和管理信息模型。

### 3.3.1 SI 满栈

Q3 接口中的 OSI 满栈包括了网络管理协议和通信协议栈（见图 3.6）。网络管理协议在 OSI 参考模型应用层的上层，有两个协议：CMIP（通用管理信息协议）和 FTAM（文件传送、存取和管理）。通信协议栈则由位于 OSI 应用层下层和 OSI 低 6 层的协议组成。具体地说，OSI 参考模型中第 1 层至第 3 层的底层协议采用 Q.811 规范，第 4 层至第 7 层的高层协议采用 Q.812 规范[15]。

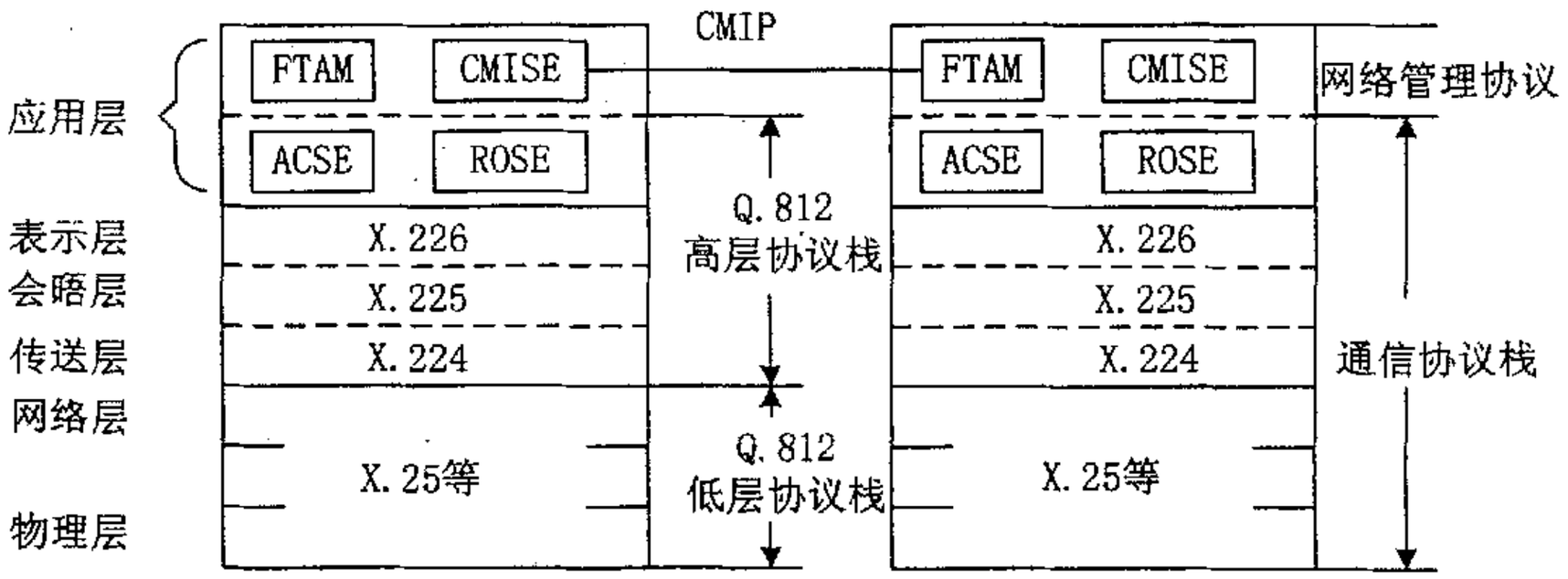



图 3.6 一个 OSI 满栈的例子

Q3 接口中的信息模型是该接口独立于设备的语义部分。

OSI 标准是一个分层的模块化体系结构，可以尽量地重用各部分的组件。在应用层，这种重用单元被称作应用服务元素（Application Service Element），简写为 ASE。不同的 ASE 就象具有不同功能的零件一样，可以根据不同需求组合成不同的实际应用。网络管理中涉及较多的 ASE 有：ACSE（联系控制服务元素），ROSE（远端操作服务元素）和 CMISE（通用管理信息服务元素）。ACSE 用于建立对等应用实体之间的联系（连接），一旦联系建立，就可以在它上面交换管理信息；CMISE 是网络管理的协议中的核心，它用于规范对等管理实体之间如何交换管理信息[16]；ROSE 的支持用来完成远端的请求一回答等操作。这样，ACSE 和 ROSE 共同为 CMISE 营造了一个面向连接的、支持远端操作的环境。

CMISE 包括服务定义和协议规范两部分，由于有了 ACSE 和 ROSE 的支持，它不再关心有关连接和远端操作的问题，而只需要定义网络管理所需要的服务原语和协议数据单元。

ITU-T X.711 定义了 7 中 CMISE 的服务原语，它们是：M-GET，M-SET，M-ACTION，M-CANCEL-GET，M-DELETE，M-CREATE 和 M-EVENT-REPORT。原语可以分为两类，一类是由管理者向代理主动发出的操作及操作响应，它们包括了除 M-EVENT-REPORT 以外的所有原语；另一类是由代理向管理者主动发出的报告事件，这就是 M-EVENT-REPORT 原语[17]。

M-GET，M-SET 用于对被管理对象的属性进行操作；M-CANCEL-GET 用于终止一个已经发出的 M-GET 原语；M-DELETE，M-CREATE 用于建立或删除被管理对象的实例；M-ACTION 用于对被管理对象执行某些特定的动作；M-EVENT-REPORT 报告在被管理资源上发生的事件，它是唯一由代理驱动的原语。

CMISE 服务原语有一些独到的特性，例如范围限定（scoping）和过滤（filtering），它们主要用于 get，set，create 和 delete 等操作，使得每次不仅可以对单个被管理对象，还可以对多个被管对象进行操作，提高了效率。

ITU-T X.712 定义了用于支持 CMISE 向上层管理实体提供服务所需的协议数据格式。所有 CMISE 服务都通过相应的 CMIP 消息来实现，其格式主要包括：用于区分不同请求的序列号，被管理资源即被管理对象的标识，包括被管理对象的类名和实例名，范围限定和过滤条件（任选），以及被管理对象的属性名和对应的值（任选）[18]。

CMIP 消息只能支持传送数据量较小的管理操作命令。在传送测量数据或计费信息等数据量较大的信息时，应采用文件的方式，以减少系统开销。所用的面向连接的协议称为 FTAM（文件传输、存取和管理）。它将发送文件转换为标准的虚拟存储器格式后发送；在接受方进行反变换，并对整个过程进行控制。

### 3.3.2 管理信息模型及其描述（GDMO/ASN.1）

对于一个完整的接口（此处指物理体系结构中的接口）定义，管理协议和信息模型缺一不可。TMN 采用了面向对象的方法来定义信息模型[19]。

前面已经提到，网络上的资源和业务被抽象成一个个被管理对象（Managed Object），简称 MO。每个 MO 具有自己的属性、通知、动作等管理参数。属性应是可管理的，即如果他们不是可控制的至少应该是可监控的。具有相通定义的 MO 被抽象为被管理对象类（Managed Object Class），因此，这些 MO 就是对应的被管理对象类的被管理对象实例（Managed Object Instance）。类和类的实例（即对象）的概念是面向对象技术中的基本概念。TMN 的信息模型也是建立在这一概念之上的。

任何信息模型都应该有标准化的描述语言规范类的定义，TMN 使用的是 GDMO，它是管理对象定义指南（Guidelines for the Definition of Managed Objects）的缩写，是 TMN 中用来定义被管理对象类以及它们之间关系的标准语言。

GDMO 中使用了属性包的概念，将多个经常在一起出现的属性定义在一个属性包中。这样，当被定义的 MO 类需要定义这些属性的时候，只需要指出属性包的名字，而不必把每个属性一一列出，它常常在多个 MO 类使用相同的一些属性时有用。这种通过属性包的方式实现的重用与通过继承的方式实现的重用是不同的，前者只是类似于 copy-paste 一级的代码级重用。当把一个属性包放在 MO 类的定义中时，这个属性包可以是必备的或可选的，如果是必备的，MO 类就必须支持属性包中的属性；如果是可选的，那么 MO 类可以支持也可以不支持属性包中的属性，由定义者指定支持的条件。

一个 MO 类的定义不仅包括属性和属性包的定义，还包括对 MO 重用包、特征包和条件包的定义。重用包提供了一种手段来重用其他MO中定义的性质。特征包由行为（Behavior）定义段、属性定义段和通知定义段组成。其中，行为定义段用自然语言来描述，类似程序代码中的注释部分。属性定义段定义了属性的有关性质，如对属性所进行的操作以及属性的重用。通知定义段则定义通知有关的属性和通知的重用。条件包由若干条件段组成，用来表示只有满足某种指定的条件时，该条件包才成为 MO 的一部分。

为保证信息交互的双方对交换的信息格式打成一致，GDMO 中定义的信息必须使用统一的文法——抽象文法记法 1（Abstract Syntax Notation One）[20]，简称为 ASN.1。ASN.1 提供了强大的跨平台表示数据的能力，并具有自解释性，即一个使用 ASN.1 编码的数据块本身包含了对自身数据的类型的解释。ASN.1 使用的编码规则基于 TLV 方式。T 代表标记（Tag），L 代表长度（Length），V 代表值（Value），它们分别给出被编码的数据的类型、字节长度和实际的值。不同的 TLV 可以嵌套。经过 ASN.1 编码的数据可以方便地映射成在网络上传输的字节流。任何平台只要掌握了这一编码原理，知道了具体编码规则，就可以无歧义地解码这些传输数据，并还原成自己的应用中可读的本地数据。

我们说信息模型是基于面向对象技术的，而面向对象技术的重要特性就是抽象、封装、继承和多态性。抽象，是把对象进行分组，提取这组对象的共同特征；封装是把一个对象的实现隐藏在所提供的服务中；继承指的是一个对象将自己的能力和行为传递给另一个对象；而多态性则表示在运行时通用一个匹配的接口，用一个对象去替代另一个对象的能力。在多态性下，具有共同描述的对象实例可以用不同的动作来响应相同的请求。MO 是网络上资源的抽象，或者说，MO 是被封装成对象的网络资源。在收到管理请求时，MO 如何进行内部操作以完成请求，对外是不可见的；同样，管理对象在发出通知之前是如何实现的，也是不可见的。MO 类之间还可以有继承关系，一个 MO 类在定义时，可以指定其他 MO 类作为父类，通常前者被称为子类（Subclass），后者被称为超类（Superclass）。一个 MO 类可以拥有多个父类（超类），MO 类之间的基础提供了类定义阶段的乃至实现阶段的重用。

网络上的资源彼此存在者复杂的关系，MO 是网络资源的抽象，这种关系也应该反映在

MO 中。一种最简单的实现方法是用 MO 中的一个属性指向相关联的 MO。

网络上的资源成千上万，相应的被抽象的 MO 也是成千上万的，那么在信息交互时，双方如何指定某个 MO 实例，这涉及到如何标识 MO 的问题。首先，要指定实例的 MO 类，然后给出该 MO 实例的名字。

标识一个管理对象类要用到 ASN.1 中的对象标识符（Object Identifier）[21]。为了区分不同的机构和应用领域那大量的对象类，CCITT 和 ISO 联合管理维护了一颗注册树，每个节点都注册在它所属的组织机构或应用领域对应的节点之下，每个上级节点可以拥有若干个下级节点。每个节点可以是组织、机构、领域、对象类或属性等。每个节点对应一个整数，用于在上级节点中唯一的标明自己。如果将一个节点从树根到它自身所经历的整数连起来，就可以唯一的标明这个注册对象。

对象标识符只是标明了 MO 所属于的 MO 类，要唯一地标识一个 MO，还需给出实例名。在 GDMO 中，通过名字绑定的方法将一个 MO 类作为另一个 MO 类（可与前一个管理对象类是一个类）的从属类，这意味着后一个 MO 类的实例（即 MO）可以包含若干个前一个 MO 类的实例。MO 类的这种包含关系在实际的网络资源中有着实际的意义，例如移动通信网上的一个基站可以包含多个小区，这种包含关系反映在 GDMO 定义中时，就是小区对应的 MO 类通过名字绑定成为基站对应 MO 类的从属类。MO 类通过层层的包含关系构成了一棵包容树，这种包容关系可以指网络资源之间的从属、控制、管理、包含等物理上或逻辑上的关系。一个 MO 类可以是它自己的从属类，例如，一个名为 Equipment 的 MO 类代表网络上的物理设备，它就可以成为它自己的从属类。这意味这一个 Equipment 的实例（物理设备）可以包含其他的 Equipment 的实例（物理设备），如同一台交换机的机架设备可以包含板卡设备一样。

每个 MO 实例可以在它所从属的 MO 实例中拥有一个名字，以便在这个 MO 实例的众多从属实例中唯一地区分自己。准确地说，这个名字实际上是一个属性值断言（Attribute Assertion）。一个 MO 被选中，当且仅当这个属性值断言为真，这个用于在上级 MO 中选择从属 MO 的属性就像是关系型数据库中的一个表中的主关键字一样，使用属性值断言选择 MO 就像在 SQL 中用 SELECT 语句的 WHERE 子句为真来选择字段。我们把这个在上级 MO 中区别从属 MO 的识别名字叫做相对识别名（Relative Distinguished Name）。一个 MO 在包容树中从树根到它自己的一串相对识别名形成了全局唯一标识这个 MO 的全局名，这个全局名叫做识别名（Distinguished Name）。

系统中的 MO 按照包容树的结构形成了一颗管理信息树（Management Information Tree），简称 MIT。管理信息树可以看作是包容树在实际系统中的实例，它构成了系统中 MO 的全部信息。这些 MO 的集合也被称作管理信息库（Management Information Base），简称 MIB。管理信息树为前面所提的范围限定提供了可能，一次 Get 操作可以同时获取十几个 MO 的属性，而不是只能获取一个。为达到对多个对象操作的目的，需要在操作中指明希望操作施加的对象。方法是给出一个基本 MO，然后用范围限定来指定操作施加的对象的范围，ITU-T 中规定了 4 种范围限定方式：基本 MO 自身；从基本 MO 到它在管理信息树中的向下 n 层；从基本 MO 开始的、在管理信息树中的第 n 层；最后一种方式是以基本 MO 为根的整个管理信息子树。

### 3.4 TMN 的管理业务

TMN 提供的管理业务是从使用者对网络进行操作、管理和维护等需求出发来描述的。每一种标准化的管理业务通过若干个管理功能组来实现，每一组 TMN 管理功能集又包含一系列 TMN 管理功能。

根据不同的电信网络和业务，ITU－T M.3200 规范确定了 13 种被管理域（Managed

Area)。它们是：电话交换网，移动通信网，数据交换网，智能网，No.7 公共信道信令网，窄带综合数字业务网，宽带综合数字业务网，专用可重新配置电信网，电信管理网（TMN），IMT-2000，用户接入网，传送网以及基础设备[22]。同时，还规定了 11 种 TMN 业务。它们是：客户管理，网络提供管理，人力资源管理，资费、计费和帐务管理，服务质量和网络性能管理，业务量测量和分析管理，业务量管理，路由管理，维护管理，安全管理以及后勤管理[23]。

如果把被管理域作为列，管理业务作为行，就得到一张具有 13 列和 11 行的二维表格。行、列的交叉处用来标明相应列中的某一电信管理域是否需要对应行中的管理业务。例如，电话交换网需要 11 种管理业务；传送网则不需要资费、帐务管理业务，也不需要路由选择和号码分析业务，共需 9 种管理业务；而接入网只需要 7 种管理业务[24]。在此基础上，需要针对每一种管理业务，确定它们在电信运营公司中所有的业务管理流程。

# 第四章 3G 移动网管系统的整体规划

根据 TMN 的逻辑分层体系结构，电信网管的管理层次分为五层，从低到高依次为网元层、网元管理层、网络管理层、业务管理层和事务管理层。本章主要讨论如何结合 3G 网络的特点和建设初期的需求，在 TMN 的基础上，构建一个从网络管理层至网元层的管理框架。并重点讨论网元管理层的实现架构。

## 4.1 3G 网络的特点及其对 3G 移动网管系统的要求

第三代移动通信系统的概念是 1985 年由国际电信联盟在模拟移动通信刚刚发展的时候首先提出的，当时被称为未来公众陆地移动通信系统(FPLMTS，Future Public Land Mobile Telecommunication System)随着时间的推移，第三代移动通信的要求和目标愈加清晰，由于 FPLMTS 这个名称含义不准确，发音拗口，1996 年国际电信联盟正式将其更名为全球移动通信系统 IMT-2000，俗称 3G。意即工作在 2000MHz 频段，预期在 2000 年左右商用的系统。1992 年世界无线电大会在 2GHz 频段分配了 23OMHz 的频率给 IMT-2000 陆地和卫星业务[25]。

相比之前的系统，3G 系统可以提供更高的接入速率，更清晰的语音质量，随时随地访问 Internet，多元化的业务应用，业务的个性化，定制业务，多媒体化的业务，Always Online……等等诸多特性。3G 在兼容 2G 的同时，本身的体系结构也发生了很大的变化。一个 3G 网络中包括[26]：一个或多个接入网络，可能使用不同的接入技术(GSM、UTRA、DECT、PSTN、ISDN……)；一个或多个核心网络，服务类型不定(GSM、UMTS、ISDN、IP、ATM……)；一个或多个智能节点网络，用于逻辑和移动性管理(IN GSM……)；一个或多个传输网络(PDH、SDH 等)，使用不同的拓扑结构(点对点、环、点到多点……)，及不同的物理介质(电磁波、光纤、铜线……)。

3G 网络自身的特点，加上人们对 2G 网管系统建设中出现的种种问题的经验总结，我们认为 3G 的网络管理系统应该具有以下特点：

(1) 具有管理同一系统中不同厂商设备的能力：一个 3G 网络中的网元设备不可能来自同一家厂商，因此作为一个网络管理系统，必须具备管理不同厂商设备的能力。2G 网管系统中就存在严重的管理分割现象，网络不能作为一个整体被管理，网管的效用大打折扣。

(2) 能够减小网络管理的复杂性：3G 网络中的网元异常复杂，3G 网管系统必须能够减少网络管理的复杂性，TMN 的分层管理思想为我们的实现提供了一个很好的思路。

(3) 能够在网元与网管之间以及网管系统自身之间通过标准接口进行通讯：网管系统之间的接口标准化的工作已经取得成效；网元与网管接口的标准化由于涉及到各厂商的利益，因此进展不大。

(4) 能够降低网络管理的费用以减少其在网络运营成本中的比重：在能确保网络高效运营的情况下，尽可能的减少成本，是每个提供商和运营商所追求的目标。

(5) 提供灵活配置能力允许新业务迅速开展：3G 网络的功能不会一成不变，作为 3G 网管也应该能够适应这种变化，并能够灵活的对新业务进行管理。

(6) 能够在费用较低的情况下平滑扩容：扩容包括功能的扩容和规模的扩容。要实现平滑的规模扩容，网管系统本身就必须具有跨平台的功能，减少工作平台迁移所带来的新的开发费用。

(7) 通过远程维护减轻维护负担：网管系统必须支持远程维护。基于 Web 的网管系统也是当今网管研究的一个重要方向。

(8) 提供综合故障管理功能：网络中故障往往具有相关性，孤立地看某个网元上的故障

很难真正解决问题。综合故障管理可收集网络中故障的相关性，提取故障知识库，有利于对网络故障作出及时有效的判断。

(9) 提供良好的安全管理功能：安全管理是网管系统的重要功能，对于一个商业运行的电信网络，网管系统的安全性和网元的安全性是提供给客户正常服务的基本保障。

(10) 能够提供好的性能管理功能使运营者能够很好地规划网络：网管系统收集的性能数据应成为网络规划系统和网络优化系统进行分析的数据源。2G 网络中收集的性能数据也可用来指导 3G 网络的规划和建设。

(11) 允许网络运营商/业务提供者存在互操作以便进行管理和计费信息交换：3G 网络中一个鲜明的特征就是会出现大量独立于运营商的服务提供商，3G 网管系统必须考虑这一新增角色对网络管理所带来的影响。

(12) 可伸缩并可用于较大和较小的配置：满足不同客户的需求，提高系统的可用性。

(13) 能够吸收 IT 和数据通信行业的新技术和标准。

(14) 对所有的网元进行统一管理(包括核心网、接入网)：在传统的 2G 网管中，由于接入网和核心网的巨大差异，往往是分别进行管理；在 3G 网络中实现统一管理，有利于以后上层网管功能的实现。

(15) 对所有的网络进行统一的管理(包括 GSM、CDMA，WCDMA 等)：在网络建设的过渡期内，必然会出现多种网络共存的局面，不同网络之间是有数据交换的。实现对不同网络的统一管理，将能更好的协调网络之间的关系，提高网络的整体服务质量。

(16) 能够保证网络的一致性：由于 3G 网络中的子网、网元的复杂性，一致性检查的实现将比 2G 网管中更复杂。

## 4.2 3G 移动网管建设目标

3G 基本网络的建设是一个逐步覆盖、逐步升级、逐步完善的过程，3G 移动网管的建设亦必须遵循这条原则。网管建设的实现目标分为近期目标和远景目标。

3G 网管系统建设的近期目标是：3G 移动网管的建设必须跟 3G 网络的建设保持同步。在短期内，网管系统必须能够实施对网元及网络服务的监控，特别是处于 2G、2.5G 向 3G 过渡期内，能够保证对几种不同网络的统一管理[27]，能够保证不同网管系统之间的正常交互。一般来说，在建设期内，移动服务提供商主要关注网络管理层及其以下层次的功能实现。这也是本文关注的重点。

3G 网管系统建设的远景目标是：当通信网络建设趋于稳定，进行有效的业务管理和商业管理就显得比较重要，这方面目前主要是一些专业的网管提供商在进行一些尝试，也没有一套完善的标准。最终实现的目标就是将整个网络的运营、管理、维护及提供(OAM&P)都置于一套完善的网络管理系统之下，从而真正实现 TMN 中描述的综合管理的功能。

## 4.3 3G 移动网管需具备的功能

3G 网管除了要提供OSI规定五个功能，在实际的实现中往往还添加了一些其他的功能[1]，其中比较重要的是软件管理和拓扑管理[28]。

### 4.3.1 软件管理

软件管理就是对网元的软件版本和运行数据进行统一管理。包含网元软件版本管理和网元数据备份两个部分：网元软件版本管理是指网管系统能够存放所有网元的多个版本信息、版本文件，用户可通过网管系统把某个版本传送到指定的某个网元或一批网元上，并激活该版本；网元数据备份是指网管系统能够备份所有的网元的运行数据，包括配置、性能、告警、

日志等。

软件管理对于网元的管理，特别是 NodeB(基站)的管理具有很大的意义：一个移动网络中往往有很多个 NodeB，当需要对 NodeB 进行升级时，就可以直接通过网管系统对一批或者全部的 NodeB 实施升级，从而避免了对网元进行一对一的升级，减轻了工作量。

### 4.3.2 拓扑管理

拓扑管理是指对网络中的网元进行的可视化管理，包括：创建、修改、删除拓扑对象，查看全网设备状态，查找网元，网元批操作，网元显示过滤，进行网元位置的精确定位等等。

## 4.4 3G 移动网管系统实现方案

一般来说，从网络管理层到网元层之间的层次结构有两种，如图 4.1 所示，a 是 NML 通过 EML 间接管理网元，b 是 NML 直接管理网元[29]。根据目前移动网络的特点，一个网络中的设备以及服务的提供者往往都不止来自于一个厂商，虽然各个厂商都声称其网管系统可以管理其他厂商的设备，但前提是其他的设备提供商需要开放其网元接口，但是从商业的角度考虑，厂商往往不愿意开放其网元层的接口[30]。另外，从 3G 发展的角度来看，在一段时期内，必然是多种接入网、核心网并存的局面，让 NML 处理这纷繁复杂的网元接口显然是不合适的。第三，为了使 NML 具有更好的可扩展性，就必须尽量减弱跟具体网元设备的相关性，通过 EML 来间接管理网元能够提高网络管理层的可扩展性。因此，NML 通过 EML 来管理网元是一个比较可行的方案。EML 由各厂商提供，向上给 NML 提供标准的接口，向下利用内部接口管理本厂商提供的网元。

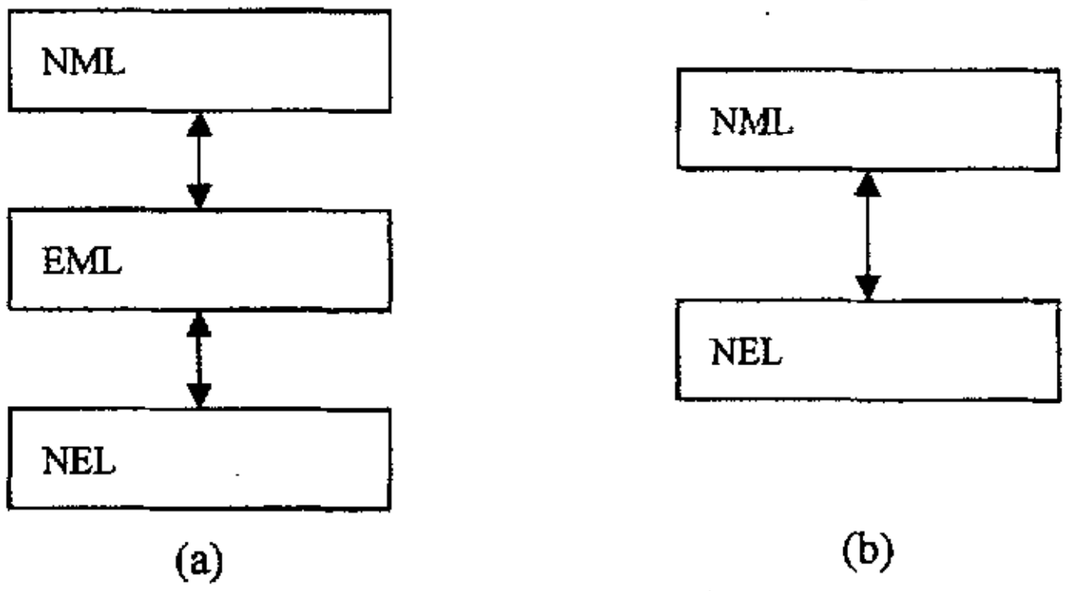



图 4.1 网管系统实现的两种方案

### 4.4.1 网络管理层与网元管理层的接口

TMN 中定义的 OSF 之间的接口都是 Q3 接口[31]。但由于 Q3 接口存在的种种问题，OMG 建议采用 CORBA 作为 NML 以上的接口，以代替实现起来非常复杂的 Q3 接口，现在甚至许多的运营商都提出需要在 EML 提供 CORBA 接口。用 CORBA 代替 Q3 是网管系统发展的方向。

Q3 独立于设备的语义描述、采用面向对象的方法编写的接口技术，它的语义和语法分开，操作和通信分开，较好地解决了网管系统的互操作问题。Q3 接口的内容包括语法和语义两部分，语法部分是指用于传达信息的通信协议和网络管理协议；语义部分则是指所传信息的内容，即信息模型。只有众多厂商的设备都遵从这些协议，并通过标准的接口交换一致性的管理信息，整个网络管理系统的协调才能得以实现。

Q3 接口的语法部分包括通信协议栈和网络管理协议：通信协议栈定义了 Q3 接口中使用的通信协议。其中应用层、表示层和会话层协议有统一的标准；而底层的通信协议没有统一的标准，在实际应用中可以选择 OSI 的 Q.811 或 Q.821，也可以选择 TCP/IP 或者 ATM AAL-5 协议[32]。在应用层规定了 3 种面向对象的服务和协议，分别是公共管理信息服务单元(CMISE)、远端操作服务单元(ROSE)和联系控制服务单元(ACSE)。公共管理信息协议(CMIP) 是 OSI 网络管理标准的网络管理协议，在 TMN 以及 SDH 的网络管理标准中都采

用了 CMIP 作为应用层交换管理信息的协议，CMIP 提供给应用层的服务就是 CMISE。

在 TMN 的信息体系结构中，建议采用 CMIS 和 CMIP 作为管理通信协议，目前所有的网管开发平台都基本支持 CMIS 和 CMIP ，实现管理通信协议的统一相对来讲比较容易做到。因此，影响实现综合网管的主要因素是网管信息模型。正是由于不同设备厂商的网管系统很难遵循统一的信息模型，才使真正的综合网管系统很难实现。由此不难看出，信息模型是网络管理的核心。

管理信息模型是 Q3 接口的语义部分。管理信息模型是基于 ISO 系统管理模型，采用面向对象的方法，使用抽象的方法使语义描述独立于设备。按照 TMN 的信息体系结构，网络管理系统采用事件触发和管理者/ 代理机制，依照管理信息模型实施管理行为。管理信息模型运用抽象句法(ASN.1)和管理对象定义指南(GDMO)来对被管理对象进行描述。GDMO 包含如下 9 个模板：被管对象类模板；包模板；属性模板；属性组模板；动作模板；通知模板；行为模板；参数模板；命名约束模板。

对于专有接口而言，异构网络互连存在问题，安全性也存在问题，虽然实现时经济方便，但在大规模的异种网络管理系统中或它们之间互连是很不适合的。Q3接口虽然支持大型电信网管与许多网元之间互连，但CMIP协议相当复杂，实现起来代价较高，而且目前在网元内部直接支持Q3接口的设备并不多见。而CORBA不仅是大型异构计算机网络互连的标准，而且与Q3一样具备很强的对象建模能力：GDMO比较适合于细粒度对象的管理，因此在底层的网管系统中采用Q3接口是比较合适的；CORBA的IDL比较适合于粗粒度对象的管理，相对于GDMO而言，IDL语言非常简洁、明了，适合于NML及上层网管接口使用。CORBA的ORB之间可以通过基于TCP/IP的IIOP协议通信，实现起来代价较低。基于CMIP的OSI系统管理体系结构需要在管理者和代理之间协商共享管理知识SMK，相比之下，CORBA采用IDL描述对象的对外接口，对被管理对象的访问和管理系统内部对象之间的交互可以一致看待，CORBA的动态调用接口DII还可以在运行时刻访问被管理系统中新的对象类型。另外，采用IDL定义对外接口的组件通过ORB可以很好地支持访问和位置透明。因此，以CORBA的ORB为核心，结合目前分布式对象体系结构和电信网络和业务管理方面的相关技术，可以构造一个更加分布化、重用性强、可靠性高、可伸缩性好并且适用于整个TMN分层体系结构的新一代分布式TMN管理平台。该平台中ORB核心以外的其它组件，如分布式处理服务、公共管理服务以及一些通用管理应用都以构件块的方式存在。所有的构件块都以一致的方式对外接口，采用OMG的IDL语言描述其提供的服务[33]。

CORBA不但能够很好的替代Q3，解决Q3接口实现中存在的问题，而且具有其他方面的优势：

(1) 基于 CORBA 的网管系统具有良好的分布特性。CORBA 的公共服务，包括名字服务，事件服务等都对分布性提供了支持。传统网管系统中，管理者和代理者是固定的关系。而在基于 CORBA 的网管系统中，管理者和代理者都是分布式对象，都是具有特定接口且可以提供一定服务的软件实体，可以将若干个分布在网络上的管理者组合为统一体共同完成网管功能。CORBA 规范实现了客户与服务器的完全分属，使得基于 CORBA 规范开发的管理代理与管理器之间只要遵从相同的调用接口就可以实现开发平台的透明性，操作系统的透明性，编程语言的透明性以及运行状态的透明性，这对支持分布异构环境的计算机网络管理系统有着极大的吸引力。

(2) 基于 CORBA 的网管系统是不倚赖于协议和编程语言的。在开发这样的网管系统时，开发者主要关注的是分布式对象的接口定义及其实现，而不关心管理者和代理之间如何通信。即从开发者看来，这样的网管系统开发基本上不涉及通信细节，而传统网管开发中，通信往往是最复杂也最不可靠的环节，有了 ORB 这个的可靠的标准的通信支持，网管系统本身的可靠性也得到了很大提高。

(3) CORBA 规范本身就是面向对象的，IDL 描述的对象接口也非常接近于 Java 语言描述，OMG 组织已经采用统一建模语言 UML 作为建模语言，CORBA 规范所遵循的面向对象的设计思想和实现方法将能够贯穿网络管理系统从设计、实现、仿真、应用和维护的整个生命周期，从而使得网络管理系统具有更强的可扩展性、可重用性，方便于系统的升级改造。因此，基于 CORBA 的网管系统开发很适合采用面向对象技术，再加上 CORBA 本身提供的大量的开发工具支持，基于 CORBA 的网管系统的组织和开发比传统的网管更有规律，更规范，也更容易些。

(4) CORBA 作为分布式计算的一个重要技术，有异构集成的巨大优势。基于 CORBA 技术构建网管系统，不仅能缩短开发周期，减少开发风险，而且利用网关实现 CORBA 网管系统和原有的 CMIP 和 SNMP 网管系统之间的无缝连接，可以再保证原有系统继续使用的情况下，实现向新的网管系统的平稳转化，推动网管系统的统一化。

另外，网管系统的的许多功能可以依靠CORBA服务来实现[34]。例如，为了提高CORBA网管系统的安全性，可以基于一个成熟的CORBA安全服务产品来实现。这些正是CORBA作为一个面向对象系统所具有的特点：封装性、继承性和健壮性所带来的天然优势。

虽然利用 CORBA 技术实现 OSF 之间的接口已成为当今网管发展的主流，但在具体实现过程中还必须注意几个问题。

1．CORBA 产品一致性的问题

目前市场上存在的 CORBA 产品种类繁多，但是由于 OMG 本身不存在对 CORBA 一致性认证的机构，因此 CORBA 产品与 OMG CORBA 规范一致性问题无法得到认证，ORB 的互连互通存在着不完备性。

2．基于 CORBA 技术的管理接口的复杂度问题

既然我们要用CORBA技术实现代替Q3接口，那么在语义上应该等同于基于Q3接口技术的实现。由于IDL其自身的简洁性，只适合粗粒度对象的建模；如果使用IDL进行细粒度对象的建模，就必须重新引用或定义大量的CORBA服务。这些服务的规定使得CORBA接口的实现变得极为复杂。因此我们必须认识到CORBA作为网管接口其自身的局限性。在本系统中，EML与网元层之间的接口就没有采用CORBA实现[36]。

3．平滑过渡的问题

由于 Q3 接口作为电信管理网络中的标准接口已有很长一段时间，目前许多已有的网元管理系统向上提供的还是 Q3 接口。为了能够实现网管系统的平滑过渡，保证网络管理系统对多种网络的管理，CORBA/Q3 网关的研究也成为网络管理研究的一个热点问题。由于 CORBA/Q3 网关不是本文研究的重点，相关资料可参考文献。

### 4.4.2 网元管理层与网元层的接口

相比较 NML 与 EML 之间的接口，EML 与网元层的接口则成熟的多。可选的接口有私有接口、Q3 接口、SNMP 接口、TL1 命令接口等。在本系统中我们采用 Manager/Agent 技术：Manager 与 Agent 之间采用私有接口；Agent 与网元绑定，Agent 与网元之间可以根据网元的性质采用 SNMP 接口或 TL1 命令接口等。具体的实现策略见下文。

## 4.5 3G 移动网元管理系统实现方案

网元管理系统 EMS(Element Management System，是网元管理层的一个实现系统)是整个系统中承上启下的一层，系统本身要实现 OSI 中定义的五种功能以及其他的一些功能；向下要负责各种网元的接入；向上要为网络管理系统 NMS(Network Management System，是网络管理层的一个实现系统)提供 CORBA 接口。

### 4.5.1 系统需求

为了达到 4.2 节中提出的 3G 移动网管系统实现的近期目标，EMS 除应完成正常的功能外，还应满足以下要求：

(1) 为了满足不同层次客户的需求，系统应做到平台无关性：即可以在高端的 UNIX 服务器上实现，也可以在普通的 Windows NT 环境下运营。这样就可以做到一次开发，多次应用，避免了为运行平台的迁移而新增开发成本。

(2) 系统能够实现平滑扩容：3G 业务是逐步丰富的，平滑的扩容能够方便对新增功能进行适配开发，并很方便的将新增模块加入到原系统中去。

(3) 系统应该实现对所有网元的动态管理(包括 GSM 网元，CDMA 网元，WCDMA 网元等)：网元的动态管理能够极大的提高 EMS 的灵活性，方便网元的接入。在网络建设初期，通信标准的变化、新功能的需求，都可能导致频繁的新增或改变网元。因此，网元的动态接入能够将这种变化带来的开发成本最小化。

(4) 系统应该实现所有功能的集中管理：EMS 相比较单个网元的管理系统最大的特点就是实现了网元的集中管理，在实现网元集中管理的基础上进行功能的集中管理，有利于管理者对网络的运营情况作出一个整体的评价。特别对于告警等功能，集中管理能更好的反应网络的真实情况。

(5) 系统应能够向不同的上层网管提供标准的 CORBA 接口：EMS 必须向上层网管提供标准接口，在上文我们已经讨论了 CORBA 接口代替 Q3 接口的原因和实现。在本系统中，我们要求能够向不同的上层网管提供满足不同需求的标准的 CORBA 接口。

### 4.5.2 关键技术

#### 4.5.2.1 中间件技术

中间件是位于操作系统和应用软件之间的一个软件层，它向各种应用软件提供服务，使不同的应用进程能在屏蔽掉平台差异的情况下，通过网络互相通信。本系统采用了如下的中间件：

##### 4.5.2.1.1 ACE(ADAPTIVE Communication Environment，自适应通信环境)

3G 网络的建设要面向不同层次用户的需求，网管系统也不例外。对于一个象中国移动这样的运营商，其一套 EMS 可能需要管理成百上千个网元；对于一个较小国家的运营商，一套 EMS 也许只要管理几十个甚至十几个网元。面对这些不同层次的需求，其硬件的配置肯定也不一样：前者可能需要 SUN Solaris 的高端服务器，后者或许用 Windows NT 就能满足需求。

我们的网管系统应该具有这种在多种环境下运行的能力。当然，我们开发的重点是如何构建出一个好的网管，而不是如何设计出一个跨平台的系统。因此，使用当前比较流行的中间件技术是一个比较好的选择。

ACE 就是我们选择用来屏蔽底层相关性的中间件。ACE 是美国华盛顿大学开发的一套主机基础设施中间件，ACE 体系结构包括三个基本层次：ACE OS Adaptation 层、ACE C++ Wrapper Façade 层和 ACE Framework 层[36]。ACE OS Adaptation 层封装了原始的、基于 C 的 OS API，隐藏了“和平台相关”的细节，展示了统一的 OS 机制接口；ACE C++ Wrapper Façade 层位于 ACE OS Adaptation 层之上，并提供了大致相同的功能，这些功能被包装成 C++类；ACE Framework 层是多组集成在一起、相互合作的类，用来为一组相关应用提供可复用的软件架构。

### 4.5.1 系统需求

为了达到 4.2 节中提出的 3G 移动网管系统实现的近期目标，EMS 除应完成正常的功能外，还应满足以下要求：

(1) 为了满足不同层次客户的需求，系统应做到平台无关性：即可以在高端的 UNIX 服务器上实现，也可以在普通的 Windows NT 环境下运营。这样就可以做到一次开发，多次应用，避免了为运行平台的迁移而新增开发成本。

(2) 系统能够实现平滑扩容：3G 业务是逐步丰富的，平滑的扩容能够方便对新增功能进行适配开发，并很方便的将新增模块加入到原系统中去。

(3) 系统应该实现对所有网元的动态管理(包括 GSM 网元，CDMA 网元，WCDMA 网元等)：网元的动态管理能够极大的提高 EMS 的灵活性，方便网元的接入。在网络建设初期，通信标准的变化、新功能的需求，都可能导致频繁的新增或改变网元。因此，网元的动态接入能够将这种变化带来的开发成本最小化。

(4) 系统应该实现所有功能的集中管理：EMS 相比较单个网元的管理系统最大的特点就是实现了网元的集中管理，在实现网元集中管理的基础上进行功能的集中管理，有利于管理者对网络的运营情况作出一个整体的评价。特别对于告警等功能，集中管理能更好的反应网络的真实情况。

(5) 系统应能够向不同的上层网管提供标准的 CORBA 接口：EMS 必须向上层网管提供标准接口，在上文我们已经讨论了 CORBA 接口代替 Q3 接口的原因和实现。在本系统中，我们要求能够向不同的上层网管提供满足不同需求的标准的 CORBA 接口。

### 4.5.2 关键技术

#### 4.5.2.1 中间件技术

中间件是位于操作系统和应用软件之间的一个软件层，它向各种应用软件提供服务，使不同的应用进程能在屏蔽掉平台差异的情况下，通过网络互相通信。本系统采用了如下的中间件：

##### 4.5.2.1.1 ACE(ADAPTIVE Communication Environment，自适应通信环境)

3G 网络的建设要面向不同层次用户的需求，网管系统也不例外。对于一个象中国移动这样的运营商，其一套 EMS 可能需要管理成百上千个网元；对于一个较小国家的运营商，一套 EMS 也许只要管理几十个甚至十几个网元。面对这些不同层次的需求，其硬件的配置肯定也不一样：前者可能需要 SUN Solaris 的高端服务器，后者或许用 Windows NT 就能满足需求。

我们的网管系统应该具有这种在多种环境下运行的能力。当然，我们开发的重点是如何构建出一个好的网管，而不是如何设计出一个跨平台的系统。因此，使用当前比较流行的中间件技术是一个比较好的选择。

ACE 就是我们选择用来屏蔽底层相关性的中间件。ACE 是美国华盛顿大学开发的一套主机基础设施中间件，ACE 体系结构包括三个基本层次：ACE OS Adaptation 层、ACE C++ Wrapper Façade 层和 ACE Framework 层[36]。ACE OS Adaptation 层封装了原始的、基于 C 的 OS API，隐藏了“和平台相关”的细节，展示了统一的 OS 机制接口；ACE C++ Wrapper Façade 层位于 ACE OS Adaptation 层之上，并提供了大致相同的功能，这些功能被包装成 C++类；ACE Framework 层是多组集成在一起、相互合作的类，用来为一组相关应用提供可复用的软件架构。

#### 4.5.2.1.2 SourcePro

在 SUN 的服务器上，一般使用的数据库是 Sybase、Oracle；在 NT 上，使用的数据库则是 SQL Server。这就要求系统实现数据库系统的独立性。

SourcePro 不但屏蔽了不同数据库之间接口的差异性，也屏蔽了数据库接口的底层实现，这样就避免了开发者过于专注于数据库繁琐的接口函数的调用和安全性的保证，从而提高了开发的效率、保证了软件的质量。

#### 4.5.2.1.3 TAO(The ACE Orb)

在前面我们曾经讨论了 Q3 接口的缺点以及如何使用 CORBA 替代 Q3 接口，其实 CORBA 不但可以作为不同 OSF 之间的接口，也可以用来作为 EMS 系统内部实现的一个重要技术。在 3.3.1.4 节我们就阐述了 CORBA 技术应用于网管系统内部的种种好处。

TAO 是使用 ACE 提供的框架组件和模式构建的 CORBA 实时实现，包含有网络接口、OS、通信协议和 CORBA 中间件组件等特性。TAO 基于标准的 OMG CORBA 参考模型，并进行了增强的设计，以克服传统的用于高性能和实时应用的 ORB 的缺点。

### 4.5.2.2 CORBA 技术

#### 4.5.2.2.1 CORBA 基本概念

公共对象请求代理体系结构(Common Object Request Broker Architecture 简称 CORBA)是对象管理组织(OMG)为解决分布式处理环境(DCE)中，硬件和软件系统的互连而提出的一种解决方案。OMG 的基本目标是开发实用的分布式对象技术及其对象管理规范，建立应用系统的通用集成框架，在分布异构的环境上实现基于对象软件的可重用、可移植和互操作。OMG 于 1991 年提出了对象管理结构(OMA)，发表了对象请求代理(ORB)的技术标准。CORBA 规范是基于抽象的对象模型的分布式对象标准，它为可移植的、面向对象的分布式计算应用提供了不依赖于具体的编程语言、运行平台、网络协议的编程接口和对象模型，非常适合于分布式系统的开发和集成。图 4.2 是从开发人员的角度看 CORBA 系统的一个总体结构图。

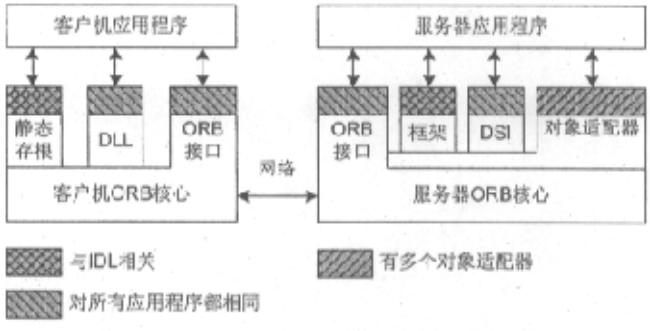



图 4.2 公共对象请求代理体系（CORBA）

如图 4.2 所示，CORBA 由许多组件组成，各个组件的功能如下所述：

(1) CORBA IDL 接口定义语言(Interface Definition Language)

在 CORBA 中，对象接口是按 OMG 接口定义语言(IDL)来定义的。IDL 一个重要的特点是独立于编程语言。因为 IDL 是一种描述性语言而不是程序设计语言，只描述接口和数据类型，不涉及实现。这就允许可以使用不同的编程语言构造对象，并且对象之间可以相互通讯。OMG IDL 文件描述了数据类型、操作和对象，客户机用它们来构造一个请求，而服务

器则为一个指定对象的实现提供这些数据类型、操作和对象。OMG IDL 支持内置的简单数据类型，比如，有符号和无符号整型类、字符类、布尔型、字符串，以及结构化数据类型，像枚举、结构、联合、序列(一维向量)和异常。

(2) 客户程序与服务器程序

客户程序是一个实体，它通过对象引用调用 CORBA 对象的一个请求，客户程序对远程对象的调用是透明的，不需要关心对象运行的物理位置及如何实现等问题。服务器程序是一个拥有一个或者多个 CORBA 对象的应用程序，实现了 IDL 中描述的接口功能，它可以用多种程序语言来完成，包括常用的 C、C++、JAVA 等语言。

(3) 静态存根和框架

当客户/服务器间的 IDL 定义好后，编译 IDL 后在客户端产生静态存根(又叫静态桩)，在服务器端生成静态框架，该静态存根负责将客户端的具体语言操作请求转变为一特定格式，经 ORB 传输到服务器端。在服务端，由 IDL 编译生成的程序框架负责将特定格式转变回具体语言操作请求，从而激发服务端相应的操作处理，操作完成后，将结果返回给调用者。在这种方式下，开发人员一旦完成 IDL 定义，便只需关注服务器相应操作的实现，根本不用关心 C/S 间的交互如何实现，大大简化工作量。这也是本系统所采用的方式。

(4) DII 和 DSI

动态调用接口 DII(Dynamic Invocation Interface)允许客户直接使用 ORB 提供的请求(Request)发送机制。不需要把 IDL 编译成的 IDL 静态存根连接到程序中，客户程序就可以通过 DII 动态地调用远程对象。动态框架接口 DSI(Dynamic Skeleton Interface)是在对象实现侧与 DII 等同的一种机制，它使 ORB 能向那些在编译时刻并不知道的对象实现动态地发送请求。

(5) 对象请求代理(Object Request Broker，简称 ORB)

ORB 是一个在对象间建立客户 / 服务器联系的中介。使用 ORB，客户可以调用服务器的对象，被调用的对象不要求在同一台机器上。由 ORB 负责进行通信，同时 ORB 也负责寻找适于完成这一操作的对象，并在服务器对象完成处理后返回结果。ORB 提供发送客户的请求到目标对象并将应答传给客户的透明通信机制，从而实现分布式处理机制。

(6) ORB 接口

ORB 接口是一种逻辑上的结构，它可以通过一个或多个进程的方式实现，也可以通过一组库函数的方式实现。为了减少应用与实现细节的相关性，CORBA 规范定义了 ORB 的抽象接口，这个接口提供了将对象引用与字符串相互转换等辅助函数。

(7) 可移植对象适配器 POA(Portable Object Adapter)

一个对象适配器是一个对象，实现请求到服务的调度。在一个服务程序应用中，POA 负责创建对象引用、激活对象以及将对各个对象的请求调度到它们各自的伺服程序上。POA 中有三个关键实体：

对象引用

对象引用标识一个唯一的对象。该对象引用提供对象的配置信息，如该对象所在的服务、服务所在的主机、以及对象所在服务中的唯一标识。客户机通过获得对象引用在 CORBA 对象上执行相应的操作。

对象标识符

POA 通过它的对象标识符识别每一个对象。对象标识符被赋值为 ObjectId 类型，这个类型在 PortableServer 模块中定义为 octet 的一个序列。

- 伺服程序

在 C++中，伺服程序是 C++对象的实例。通常，它们由派生于编译 IDL 接口定义时所产生的框架类所定义。为实现操作，可以重载框架的基本类的虚拟函数。用对象适配器注册

这些 C++伺服程序，以便当客户机调用由这些伺服程序具体化的那些对象的请求时，允许对象适配器调度请求给这些伺服程序。

#### 4.5.2.2.2 CORBA 服务

(1) CORBA 命名服务

OMG 命名服务是最简单也是最基本的 CORBA 服务。它提供从名称到对象引用的映射：给定一个名称，该服务返回一个存储在此名称下的对象引用。

在网管系统中，MO(Managed Object)由对象工厂创建，一旦对象被创建成功，一个对象引用即刻产生，并作为操作结果的返回值。之后该引用可以作为参数传递，被其他应用执行相应的操作。如图 4.3，对象工厂创建对象后，触发命名服务的绑定操作，登记名称到对象引用的匹配关系。其他应用可以根据名称向命名服务查询，以获得对象引用。

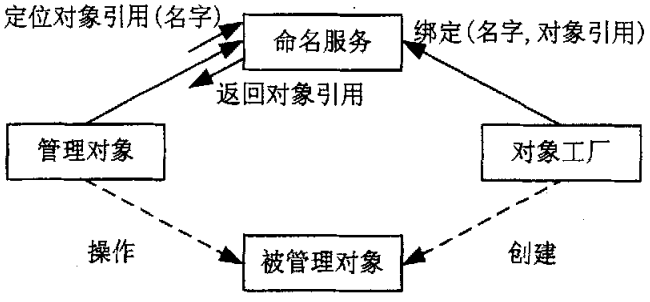



图 4.3 使用命名服务定位对象图

(2) CORBA 事件服务

在同步请求机制下，一个主动的客户程序向被动的服务器调用请求；在发送一个请求之后，客户机阻塞并等待返回结果。许多分布式应用程序都发现，虽然同步请求调用很有用，但是它限制太严格。

OMG 事件服务允许应用程序使用解耦通道通信模型而不是使用严格的客户机-服务器同步请求调用机制。事件服务允许提供者在一次单个调用的情况下向一个或者多个用户发送消息。事实上，使用事件服务实现的提供者不需要知道消息的用户是谁；事件服务在提供者和用户之间起中介的作用。

在 OMG 事件服务(Event Service)模型中，提供者(suppliers)生成事件(events)而使用者(consumers)接收事件。提供者和使用者都连接在一个事件通道(event channel)上。事件通道将事件从提供者传送到使用者，而且不需要提供者事先了解使用者的情况，反之亦然。事件通道在事件服务中起着关键作用，它同时负责提供者和使用者的注册，可靠地将事件发送给所有注册地使用者，并且负责处理那些与没有响应地使用者有关的错误。

CORBA 标准定义了两种事件传输模式，一种称为事件服务，另一种称为通知服务。由于事件服务不支持结构事件定义和过滤，因此本网管系统事件服务是基于通知服务的。以下为 CORBA push 事件模型：

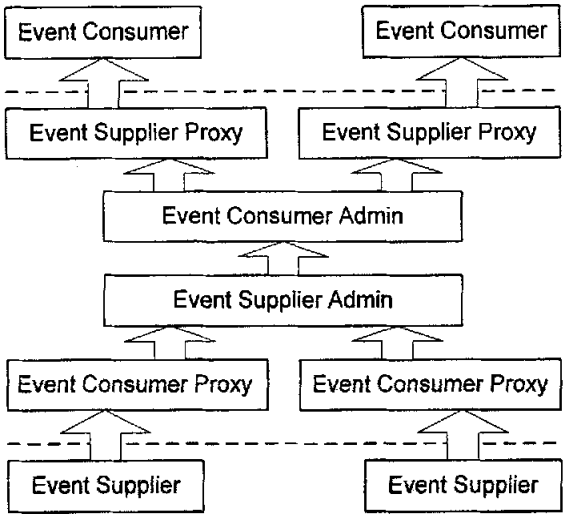



图 4.4 CORBA 通知事件模型（push 模式）

事件使用者选择相应的事件接收通道，通过事件使用者，Amdin 对象创建事件提供者代理。通过事件提供者代理，代理对象设置过滤条件并创建两者间的连接。

事件提供者选择相应的事件发送通道。通过事件提供者，Amdin 对象创建事件使用者代理。通过事件使用者代理，代理对象设置过滤条件并创建两者间的连接。

事件提供者发送事件到相应的事件使用者代理对象，如果该事件符合设置的过滤条件，事件将缓冲到事件队列，事件发送流程结束并返回。

对于事件队列中缓存的所有事件，检查所有事件提供者代理对象，如果事件符合事件提供者代理对象中设置的过滤条件，那么该事件将发送到相应的事件使用者对象。

#### 4.5.2.2.3 电信网应用 CORBA 进行管理的优势

CORBA 技术适合电信网管理应用主要是由于 CORBA 的技术特点决定的。目前电信部门在设计它的 IT 系统时所面临的主要问题是电信业务的分布性和大规模扩展能力。为提供远距离或全球范围的通信，电信部门之间必须进行有效集成和互操作。另外，电信公司还需要不断地提供新的服务，如视频会议、Internet 技术、点播服务或交互式服务等，同时保护原有系统的投资（原有数据、应用等）。

CORBA 之所以非常适合电信领域主要有两方面的因素：一是 CORBA 的技术特点，如采用先进的软总线/软构件的层次结构和面向对象技术，容易实现遗留系统的集成，符合标准的处理流程，系统的开放性适应新技术、新业务的发展；二是电信领域的需求特点，即超强的分布处理需求，而这正是 CORBA 的优势所在。

电信网应用 CORBA 进行管理的优势可概述如下：

(1) CORBA 技术采用层次化的系统结构，可以使应用系统的结构更加清晰，便于实现和维护。

(2) 采用基于软构件的技术，可以进行应用的快速构造，提高系统的可靠性及快速开发能力。

(3) 采用软总线结构，不仅能够支持应用集成框架的建立，满足协同工作的需求，而且建立了多层次的软构件技术，更加便于应用领域框架及领域构件的开发，从而满足电信综合业务的快速构造和灵活部署。真正做到"即插即用"。

(4) 基于 CORBA 技术，可以方便地实现系统的可移植性、互操作性和分布透明性，方便地进行系统的扩展和升级。

#### 4.5.2.2.4 CORBA 在电信网管系统中的应用

在电信运营体系中，网管系统一直是网络维护工作的重点。随着电信运营商网络规模的不断扩大，多厂商设备环境下的网络管理是网络管理研究和网管系统建设中的难点。CORBA 技术在标准性、规范性、开放性方面的优势，为我国的网管应用水平的提高提供了一个很好的思路。从目前来看，TMN 和 CORBA 技术结合的方式是目前构建网管系统最为理想的一种解决方案。

在探讨 CORBA 在电信网管系统中的应用时，一般认为 CORBA 技术可以在以下三个方面发挥优势：

(1) 在开发网管系统的运行系统（OS）时，CORBA 可以为组成 OS 的内部功能单元间的交互提供通信方式。即利用 CORBA 软构件技术来构建网管功能服务对象，来满足网管应用的需要。同时，利用 CORBA 软总线技术达到不同构件之间的协同工作。

(2) 在不同系统之间的互操作时，CORBA 作为标准的中间件，支持与编程语言无关的接口定义。由于 OMG IDL 具有标准的语言映射和有多厂商支持的特性，因此，非常适合不同系统之间的互操作。

(3) CORBA 作为管理系统和被管资源间的通信接口。即在 OMC 层次上提供标准的 CORBA 接口，满足上层规范化管理的需要。

目前，作为设备厂商提供的网管接口（OMC 层次上）存在着很大的私有性和混乱性，究其原因，很大程度上是缺少多厂商共同遵循的规范。而目前各电信运营商在构建其网管系统时，也意识到了存在着这些异构性的问题，因此，也在积极的推进标准规范的制定和发展，并且也取得了很大的成果。比较统一的意见是，从长远角度来看，选用一种独立于具体厂家的技术，来开发多厂家环境下的网管系统是非常合理的。而 CORBA 在标准性、规范性、开放性方面的优势，为我国的网管建设提供了很好的解决方案。并且，CORBA 技术已经得到了国际电信联盟电信标准部（ITU-T）的充分认可，并制定了相关的规范，这些规范表明 CORBA 在网管中应用的基础标准化工作已经完成。在密切跟踪国际进展的同时，国内对在 TMN 中引入 CORBA 技术的准备工作已经开展了多年，并积累了丰富的实际经验，而且有些成果已经达到世界先进水平，为国内网管建设过程中采用 CORBA 技术铺平了道路。中国电信运营商及相关研究机构也根据 ITU 相关规范，制定了针对中国设备供应商所应提供的 CORBA 接口。为中国电信设备接口规范化、网管系统标准化打下了基础。

应用 CORBA 可以带来如下好处：

(1) 技术先进性

CORBA 代表着计算机信息处理的发展方向，其技术特点非常适合构建电信级应用。采用 CORBA 技术，层次架构清晰，可以更好的满足电信运营商的要求，也符合通信技术的发展趋势。

(2) 提高产品核心竞争力

采用 CORBA 技术，可以使应用软件层次结构更加清晰，满足规范性、标准性、开放性的要求，可以很方便地进行系统的扩展和升级。采用 CORBA 技术，一方面可以降低自己的劳动生产成本，另一方面可以大大提升产品的核心竞争力，从而在激烈的电信市场竞争中占据有利地位，获得更大的市场份额和更好的经济效益。

#### 4.5.2.2.5 CORBA 在 3G 移动网管系统中的应用

(1) CORBA 作为网管北向接口

经过多年的发展，TMN 已成为国际公认的全球性电信管理网框架。TMN 标准定义了一套接口，包括管理功能、通信协议和信息模型等。其中最重要的接口是与 OS 相连的 Q 接口。近年来 ITU－T 采纳了许多专家的意见，将 CORBA 技术引入电信管理网，作为 Q 接口的一

种可选方案。在过去的数年，网管北向接口一直是 Q3 接口或私有接口占主导地位。但由于 Q3 接口对象粒度过细、复杂度较高，具有过多的条件包和可选项，且接口信息交互的效率较低，开发成本高,软件的可重用性差，因此，难于真正实现不同厂家设备之间的互联互通。而 CORBA IIOP 将 TCP/IP 作为其下层传输协议，相对于 Q3 接口所采用的 OSI 7 层协议来说是一个轻量级的协议，它不存在复杂而耗时的编解码过程，因而运行的效率高很多。CORBA 接口具有简单、易实现、易修改和易扩充等优点，已经逐渐被各大厂商用来开发标准接口，并有逐渐取代 Q3 接口的趋势。图 4.5 给出了 CORBA 作为网元/子网级网管和网络级网管联接及网络级网管和业务级管理系统联接的北向接口的应用实例。

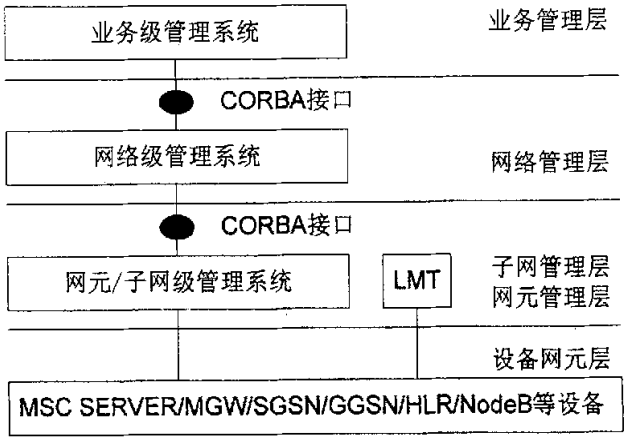



图 4.5 CORBA 作为网管北向接口实例

(2) CORBA 作为不同制式系统统一管理的总线

第三代移动网络系统采用的技术是多样化的，可以采用 WCDMA、CDMA2000 和 TD-SCDMA 等无线接入技术，核心网可以采用 GSM MAP 网络、ANSI-41 网络或者全 IP 网络，一种 CN 技术制式可以和多种 RAN 技术制式配合。为了避免针对不同技术以及不同的 CN 和 RAN 技术制式组合开发不同网管系统，并满足采用多制式的运营商的网管需求，使网管系统可以同时管理多种技术，3G 网管的网络管理环境应是一个多种技术制式并存的环境。此外目前的移动通信运营商通常采用第二代移动通信系统 2G，这些运营商在向 3G 过渡的过程中，必将出现 2G 和 3G 系统共同的局面，而且这种局面可能会存在较长时间。为使 2G 和 3G 移动通信系统能够协调运行,3G 网管系统应该能够管理遗留下来的 2G 系统。

采 CORBA 结构可以完成 WCDMA、CDMA2000 和 TD－SCDMA 3 种不同制式和遗留 2G 系统的统一综合管理，适应了 3G 网管系统的多制式环境，并兼容 2G。

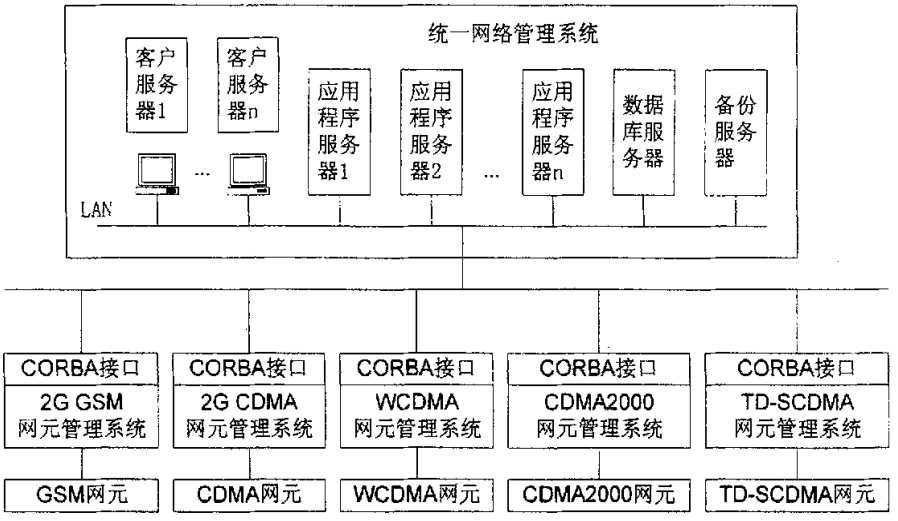



图 4.6 CORBA 作为不同制式系统统一管理的总线

(3) CORBA 作为 3G 移动网管系统内部通信平台

网络管理系统可采用模块化的方法加以设计和实现，其功能结构如图 4.6 所示。根据第三代移动通信网网络管理接口的基本功能需求，分别构建性能管理模块、配置管理模块、故障管理模块、安全管理模块、公共管理模块等功能模块。各功能模块可运行在一台服务器上，也可分别运行在不同的服务器上。各功能模块之间可通过 CORBA 接口实现数据交互，它们相互配合、协同操作，构成整个系统的应用服务层。应用服务层通过数据库服务器和数据库之间实现数据的交互。界面程序运行在多个客户终端上，界面程序和应用程序之间采用 CORBA 接口实现数据通信。界面程序和应用程序可运行在不同的操作系统之上，也可采用不同的程序设计语言加以实现。

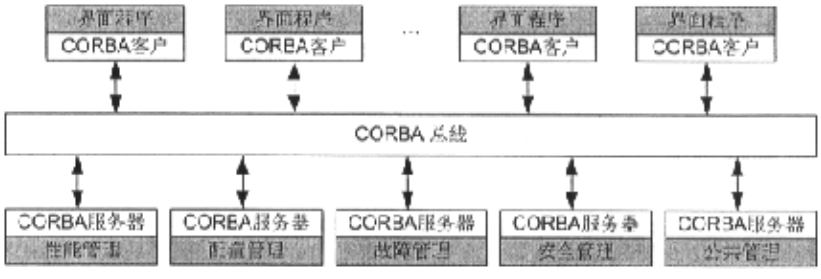



图 4.7 CORBA 作为 3G 移动网管系统内部通信平台

### 4.5.2.3 XML 技术介绍

#### 4.5.2.3.1 XML 起源

XML（eXtended Markup Language）是从 SGML（Standard Generalized Markup Language）进化来的，最初 SGML 是为了解决文档及其格式问题的一种标记语言，所以非常的复杂和难以使用，那时计算机的应用水平还处在很低的阶段，并且那时人们只需要传递和显示一些简单的数据。因此，89 年 Tim Berners-lee 依据 SGML 开发出一种超文本格式，就叫 HTML（HyperText Markup Language），非常的简单。当计算机的应用水平大幅度的提高的时候，

人们已经对太简单的 HTML 开始不满意，因此又开始对其进行修改、升级，就这样 HTML 从 1.0 一直升到 4.0 的版本，扩展了许多的命令，实际上也是为了让 HTML 解决更多的问题，但是因为 HTML 本身的缺陷，为了解决更多的问题，把简单的 HTML 变成一个非常复杂的，难以理解的语言，再加上 HTML 是把数据和显示格式一起存放的，所以如果我们只想使用数据而不需要格式，分离这些数据和格式就显得十分的困难。

随着 Internet 的发展，98 年 1 月 W3C 公布了 XML1.0 版本，立即成为计算机史上一个重要的里程碑。

XML 是一种简单、与平台无关并被广泛采用的标准，它提供一种描述结构化数据的方法。与主要用于控制数据的显示和外观的 HTML 标记不同，XML 标记用于定义数据本身的结构和数据类型。XML 相对于 HTML 的优点是它将用户界面与结构化数据分隔开来。这种数据与显示的分离使得集成来自不同源的数据成为可能。客户信息、订单、研究结果、帐单付款、病历、目录数据及其他信息都可以转换为 XML。

#### 4.5.2.3.2 XML 文档

一个完整的 XML 文档内容一般包含下面四个方面：处理指令、元素、属性和注释。

(1) 处理指令

处理指示是用来给处理 XML 文件的应用程序提供信息的。也就是说，XML 分析器可能对它并不感兴趣，而把这些信息原封不动地传给 XML 应用程序。然后，这个应用程序来解释这个指示，遵照它所提供的信息进行处理，或者再把它原封不动地传给下一个应用程序。所有的处理指示应该遵循下面的格式：

<?处理指示名 处理指示信息?>

XML 声明是处理指示的一种，它的作用就是告诉 XML 处理程序："下面这个文件是按照 XML 文件的标准对数据进行置标的"。一个完整的 XML 声明是这样的：

<?xml version = "1.0" standalone = "no" encoding = "GB2312"?>

- version 属性

在一个 XML 的处理指示中必须包括 version 属性指明所采用的 XML 的版本号，而且，它必须在属性列表中排在第一位。由于当前的 XML 最新版本是 1.0，所以我们看到的无一例外的都是：version = "1.0"。

- standalone 属性

这个属性表明该 XML 文件是否和一个独立的置标声明文件配套使用。因此，如果该属性置为"yes"，说明没有另外一个配套的 DTD 文件来进行置标声明。相反，如果这个属性置为"no"，则有可能有这样一个文件。（注意，也可能没有。）

- encoding 属性

所有的 XML 语法分析器都要支持 8 位和 16 位的编码标准。不过，XML 可能支持一个更庞大的编码集合。在 XML 规范的 4.3.3 节中，列出了一大堆编码类型。但我们一般用的有简体中文码（GB2312）、繁体中文码（BIG5）、西欧字符（UTF-8）。采用哪种编码取决于文件中用到的字符集。

XML 声明中要求必须指定"version"的属性值，其余两个可选。

(1) 元素和属性

元素是 XML 文件内容的基本单元。从语法上讲，一个元素包含一个起始标记、一个结束标记以及标记之间的数据内容。我们把 XML 中开始和结束标记之间的文字称作"字符数据"，而把置标内的标示文字称作"标记"。有些元素的字符数据为空，称它们为空元素。属性是元素的性质，标记包含元素和属性两部分，如下所示：

<标记名 属性名称="属性值">字符数据</标记名>

每个 XML 文档都必须有一个且仅有一个包含其他所有元素的元素，这个囊括一切的元

素称为根元素或者文档元素。根元素紧跟在 XML 声明之后，如果该 XML 文件中还内嵌了 DTD，则这个根标记紧跟在 DTD 之后。根元素必须是一个非空的标记，其中包含了整个文件的数据内容。在 HTML 中，根元素是<HTML>，所有其它元素都在打开和关闭 HTML 标记的约束内。在 XML 中，所有 XML 文档都不以<XML>开始，因为字符串 XML 是保留字，除此之外可以自由地创建自己地根元素。

(2) 注释

注释在几乎所有的标记语言或者程序设计语言中都可以出现，以备自己或其他读者日后参考。在 XML 中书写的注释句法等价于 HTML 和 SGML 中的注视：它以标记说明打开界定符（<!）和注释打开界定符（--）成组合界定符<!--，然后以-->束。XML 处理器对于注释中的一切内容都会视而不见，注释中出现的标记也一同被忽略。

#### 4.5.2.3.3 XML DTD 和 Schema

在 XML 中，“形式良好”（well-formed）有着明确的标准，即是要遵守 XML1.0 规范中的语法规则。无论是从物理结构上讲，还是从逻辑结构上讲，XML 都必须符合规范，才能被正确解释处理。但 XML 文档仅仅满足良构还不够，它必须是有效（valid）的，即对一个 DTD（或 XML Schema）有效，这意味着 XML 分析器（parser）在解释一个“有效的”XML 文件之前已经知道了所有规则的细节。

(1) DTD（Document Type Definition）文档类型说明

DTD 描述了一个置标语言的语法和词汇表，也就是定义了文档的整体结构以及文档的语法。简而言之，DTD 规定了一个语法分析器为了解释一个“有效的”XML 文档所需要知道的所有规则的细节。它主要包括以下内容：

元素类型说明：DTD 中使用元素类型声明 ETD（Element Type Declaration）来声明所有有效的文档元素。ETD 应该采用如下的结构：

<!ELEMENT 元素名 元素内容描述>

元素属性：属性使得文档作者能够提供关于元素的额外信息。与元素一样，属性必须在 DTD 中说明。每个元素的属性在属性表中说明，而不是一个一个地说明。属性表格式为 <!ATTLIST 元素名 （属性名 属性类型 缺省值）*>。其中，元素名是属性所属的元素的名字，属性名是属性的命名，缺省值是属性的初值。属性值类型有以下四种类型：串类型、标志化类型、枚举类型和实体。

实体：简单地说，实体就是一个存储容器，它能容纳从缩略地扩展定义，到可能在大型文档不同地方反复使用地大段 XML 文档。它地类型一般有：一般实体、SYSTEM 和 PUBLIC 标识符、外部实体、不分析实体、参数实体。

(2) XML Schema

Schema 文件和其它 XML 文件的样子非常相似，有着自己的一套完整的语法。它是由一组元素构成的，涉及到的关键元素包括：Schema、ElementType、element、AttributeType、attribute、group、datatype、description。其中，Schema 是根元素。Schema 文件的元素以及属性在此不作过多描述，可以参照参考文献。

DTD 作为 XML 1.0 规范的重要组成部分，对于 XML 文档的结构起到很好的描述作用。但是，它也具有一些缺点，比如，它采用了非 XML 的语法规则、不支持数据类型、扩展性较差等等。Schema 正好解决了这些问题。从总体上讲，Schema 具有以下优点：

一致性：Schema 使得对 XML 的定义不必再利用一种特定的形式化的语言，而是直接借助 XML 自身的特性，利用 XML 的基本语法规则来定义 XML 文档的结构，使得 XML 达到了从内到外的完美统一，也为 XML 的进一步发展奠定了坚实的基础。

扩展性：Schema 对 DTD 进行了扩充，引入了数据类型、命名空间，从而使其具备较强的可扩展性。

互换性：利用 Schema，我们能够书写 XML 文档以及验证文档的合法性。另外，通过特定的映射机制，还可以将不同的 Schema 进行转换，以实现更高层次的数据交换。

规范性：同 DTD 一样，Schema 也提供了一套完整的机制以约束 XML 文档中置标的使用，但相比之下，后者基于 XML，更具有规范性。Schema 利用元素的内容和属性来定义 XML 文档的整体结构，如哪些元素可以出现在文档中、元素间的关系是什么、每个元素有哪些内容和属性、以及元素出现的顺序和次数等等，都可一目了然。

#### 4.5.2.3.4 Xerces-C++介绍

随着 XML 越来越广泛地被采用，高效解析 XML 文档也变得越来越重要，尤其是对于那些要处理大量数据的应用程序。不正确的解析会导致过度的内存消耗和过长的处理时间，从而有损于可伸缩性。

XML 解析器将一个未经处理的序列字符串作为输入，并对它执行一些特定的操作。首先它检查 XML 数据是否符合句法规则，确保开始标记与其有匹配的结束标记，并且没有重叠的元素。大多数解析器还根据文档类型定义（Document Type Definition，DTD）或 XML Schema 进行确认，核实其结构和内容是你所指定的。最后，解析输出通过编程 API 提供对 XML 文档内容的访问。

有三种用于 Java 的流行 XML 解析技术：

(1) 文档对象模型（Document Object Model，DOM），一个来自 W3C 的成熟标准。

(2) 用于 XML 的简单 API（Simple API for XML，SAX），第一个被广泛采用的用 Java 编写的 XML API，是一个事实上的标准。

(3) 用于 XML 的数据流 API（Streaming API for XML，StAX），JSR-173 中采用的一个很有前途的新解析模型。

Xerces-C++是 Apache 软件基金会开发的解析器。它的包支持文档对象模型（DOM）和 XML 简单应用编程接口（SAX）。两种解析器分别被指定为 Xerces-C++ DOM 和 Xerces-C++ SAX。以 DOM 的方式解析，看待每个 XML 文档为一个对象；然而以 SAX 的方式，看待 XML 文档为一系列的事件。Xerces 同时提供了 DOM 和 SAX，因此他们在分列类时有各自的方式。有三种文档类型处理器承担实际的解析工作。对于 DOM 解析，他们是 DOMParser 和 IDOMParser；对于 SAX 解析，有 SAXParser。实例化他们其中一个引擎是解析 XML 文档的第一步。

例如，下面的代码片段举例如何构造新的 DOMParser 对象，解析器指向它。

```
DOMParser* parser = new DOMParser;
```

创建 SAXParser 同样简单：

```
SAXParser* parser = new SAXParser;
```

本论文采用的是 SAX 解析技术。SAX 采用基于事件驱动的方式来处理 XML 文档。基于事件是指 SAX 为应用开发者提供了处理感兴趣的特定元素的方法，而不必要求在应用层次处理之前预先建立元素。SAX 这个接口规范是 XML 分析器和 XML 处理器提供的较 XML 更底层的接口。从本质上说，SAX 是一种 Java 接口，它能给应用程序提供较大的灵活性。

SAX 使用事件和方法回调。在 XML 文档中的每个节点表示为一个 SAX 事件（像开始标签或结束标签）。为了处理文档，新建一个 DocumentHandler 类的实现。文档处理器必须实现 SAX 解析器将要为 XML 文档中不同的事件调用的各种方法。例如，DocumentHandler 类中的 endElement()方法，当 end-element 事件发生时随时被 SAX 解析器调用。SAX 解析器通过注册的过程知道如何调用你的处理器。下面是使用 SAX 解析器时，如何注册处理器的一个例子：

```
SAXParser* parser = new SAXParser;
MySAXHandlerClass handler;
```

parser->setDocumentHandler(&handler);

一旦文档处理器在 SAX 解析器中被注册，就可以调用解析器的 parse()方法。这将开始为文档处理器的各种处理方法传送事件的过程。

#### 4.5.2.3.5 XML 技术在电信网管中的应用

基于 XML 的网络管理技术采用 XML 语言对需要交换的数据进行编码，为网络管理中复杂数据的传输提供了一个极好的机制。XML 文档的分层结构可以对网络管理应用中的管理者-代理模式提供良好的映射，通过 XSLT(Extensible Style sheet Language Transformations)样式表可以对 XML 数据进行各种格式的重构和转换，加上 XML 已经被广泛应用于其他领域，各种免费和商业的 XML 开发工具发展异常迅速，因此使用 XML 来定义管理信息模式和处理信息十分便利。

#### 4.5.2.3.6 网络管理应用 XML 的优势

XML 能成为网络管理中值得研究和使用的工具，必须具备一些其它网络技术所不能提供的特性，主要表现如下几个方面：

(1) 复杂数据处理优势

XML 是一种结构化数据，它简单的编码规则使得可以使用 ASCII 文本和类似 HTML 的标记来描述数据的任何层次，通过 DTD 或者 XML Schema 来定义元素的顺序和结构，DTD 和 XML Schema 提供了一种发布数据改变的正规机制。使用 XML 对比工具来比较新、旧两个 XML Schema 文件，就能得到数据的哪些特征、选项或是输出标记发生了变化的详细情况。

(2) 使底层数据更具可读性和标准型

目前网络中传输的底层数据通常根据网络协议的不同，而采用的编码规则不同，虽然最后在传输的时间都转化为二进制位流，但是不同的应用协议需要提供不同的转换机制，在洗衣所能理解的数据与二进制数据之间进行转换。这种情况导致网络管理站在对采用不同协议发送管理信息的被管对象之间进行管理时很难实现兼容性。但是如果这些协议在数据表示时都采用 XML 格式进行描述（XML 的自定义标记功能使这一需求成为可能），这样网络之间传递的都是简单的字符流，可以通过相同的 XML 解析器进行解析，然后根据 XML 标记不同，对数据的不同部分进行区分处理，使底层数据更具有可读性和标准型。

(3) 构建被管网网元模型

在使用 XML 模板构建被管网元模型，可最大限度地增强网络管理软件地灵活性和可扩展性，通过 XML 模板构建被管网元模型，可以满足网元对象模型的以下要求：

- 用最少的对象模型来描述最多种类的网元对象；
- 尽量避免特殊化，模糊各厂家产品自身的可管理性；
- 对象的层次结构无论负责还是简单，都可以用相同的数据结构来表示；
- 通过该模型能方便地得到网络管理地五大功能模块所需地管理数据；
- 有足够的可扩展空间，使得出现新的被管网元对象时，该模型同样也能适应。

(4) WEB 网络管理优势

XML 在网络管理的优势中，同样体现在基于 WEB 的网络管理方式中。

#### 4.5.2.3.7 XML 在 3G 移动网管中的应用

将 XML 应用于网络管理时网络管理领域的必然发展趋势，因此本文所讨论的 3G 移动网管系统性能管理设计和实现中，在这方面做了大胆的尝试，主要在以下几个方面运用了 XML 技术：

(1) 利用 XML 进行应用程序初始化配置。

(2) 性能数据采用 XML 文件封装，通过 FTP 传输的形式从网元上传到服务器。

(3) 导出信息支持 XML 文件格式，如在 GUI 导出任务信息。

#### 4.5.2.4 网元的动态接入

作为 EMS，网元的接入是进行网元管理的前提，也是历来 EMS 较难解决的问题。由于网元的种类众多，版本变化快，因此必须要找到一个合适的方法实现网元的动态接入，即所谓的“即插即用”。

传统的移动 EMS 采用 EMS 直接管理网元的模式，这样 EMS 就必须花费大量的精力来处理接口适配的问题，由于网元之间被管对象的巨大的差异性，直接从网元中提取出独立于设备的语义几乎无法实现。用 EMS 来处理网元接口的做法不但降低了开发效率，增加了开发风险，而且不利于产品的升级换代，更是无法实现“即插即用”。

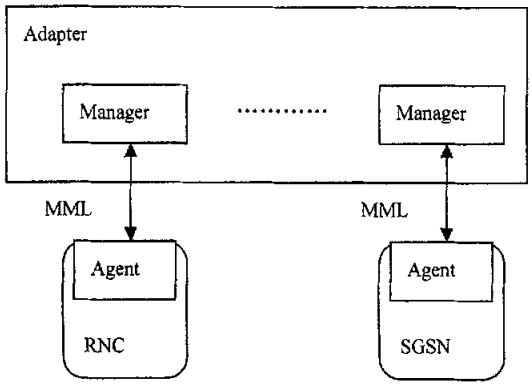



图 4.8 网元接入模型

在本系统中，我们采用 Manager/Agent 模式(如图 4.8 所示)，接口实现的细节由 Agent 处理：Agent 与网元之间根据网元的种类可采用 SNMP 接口(IP 设备)或 TL1 命令接口(普通的电信设备)；Manager 与 Agent 之间采用私有接口 MML(Man Machine Language 人机交互语言)进行交互。

(1) Agent 随网元版本发布(Agent 是网元实现的一部分，下文若无特殊说明，网元皆包含 Agent)；

(2) Manager/Agent 之间的接口遵循独立于设备的标准(MML 标准)；

(3) Manager 通过 MML 在 EMS 侧用 MIT (Managed Information Tree)维护所有网元的 MIB 信息；

(4) EMS 的所有功能模块利用 MIT 的信息进行对网元的管理。

将与具体设备相关的部分放到了网元内部处理，网管与网元的交互变成了网管与 Agent 的交互，这样就易于实现独立于设备的语义描述，MML 就是实现这种语义描述的语言。比如对于 DSP，EMS 与 Agent 之间的交互只要知道是 DSP 就行了，至于它是什么类型的 DSP，怎么进行性能数据的统计，就是 Agent 与网元之间需要关心的问题了。

实现了接口语义独立于设备，也就易于实现网元的动态接入，同时也就解决了许多开发和维护阶段的问题：

首先减少了网元接入的开发量：如果能够保持 MML 规范的稳定，网元版本发生变更或新接一个网元时，不需要进行 EMS 版本的更新；其次，方便了 EMS 和网元的升级：由于不需要过多的关注接口的问题，动态升级也就很容易实现了；第三，减少网络通信开销：EMS 与 Agent 的信息交互量要比 Agent 与网元之间的信息交互量要小的多，而 Agent 与网元是绑定的，它们之间的通信效率要比 EMS 与 Agent 之间的通信效率高很多。

当然，目前的网元动态接入也是有其局限性的：

受 MML 表示能力的限制。这是一个内部接口，这就决定了采用这种方法实现的 EMS 只能管理本公司的网元，并且当现有的 MML 无法表示新增的语义时，这种方法将不再有效。

对 EMS 的实现效率有影响。由于增加一个中间层，MML 的解析将会降低 EMS 的实现效率。只要这种影响能够控制在可以容忍的范围内，动态接入所带来的好处足可以抵消这种

影响。

### 4.5.3 实现方案

EMS 的系统结构图见图 4.9。

系统采用 CORBA 作为内部软总线，所有的功能模块通过 CORBA 进行交互：既可以通过 CORBA 的 IIOP 进行对象服务的调用，也可以通过 CORBA 提供的事件机制型进行互操作。利用 CORBA 作为软总线，不但能够实现分布式计算，而且能够实现系统功能的平滑扩

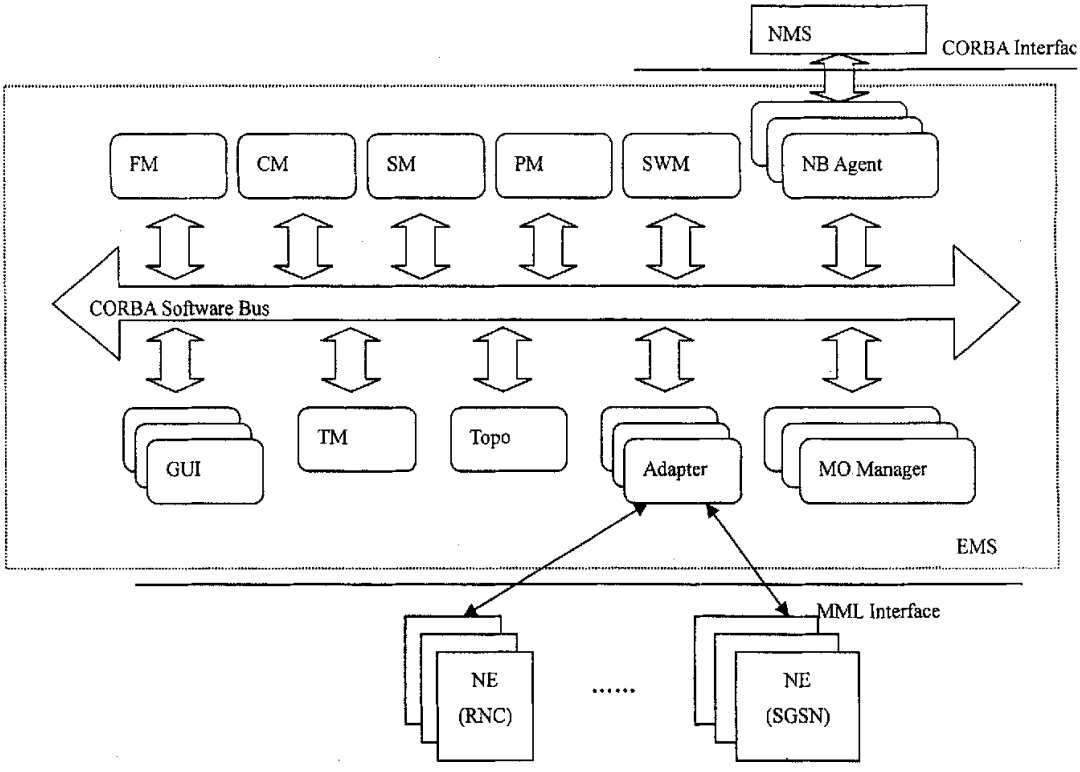



图 4.9 EMS 系统结构图

容。

系统采用 GUI/Server/Adapter 三层结构，Server 处理业务相关功能，Adapter 实现网元接入的功能：Server 层包括除 Adapter 和 GUI 外所有的模块；Adapter 模块专门来处理网元接入的问题，所有的 Manager 都在 Adapter 中实现。OSI 中定义的功能模块通过 Server 和 Adapter 协同实现。Server 与 Adapter 的分割使得 Server 层彻底屏蔽了网元相关性，使其能够专注于业务功能的开发实现。

系统通过 MO Manager 来管理 MIT：MO Manager 通过 Adapter 从网元获取相关的 MIB 信息，并写入 MIT；同时为其他模块提供对象查询接口；MIT 与 MIB 对象信息的一致性通过事件机制来保证。

系统通过 NB Agent 来实现对上层网管标准接口的提供，为了满足不同上层网管的需求，可以通过 NB Agent 进行适配开发：既不影响已有的系统实现，又满足了新上层网管的需求。在提供标准 CORBA 接口的前提下，还能灵活的为不同需求的上层网管提供 CORBA 接口。

系统的网元接入采用 4.5.2.4 节讨论的 Manager/Agent 结构，实现网元的动态接入，屏蔽不同子网、不同网元带来的网元管理的差异性。

影响。

### 4.5.3 实现方案

EMS 的系统结构图见图 4.9。

系统采用 CORBA 作为内部软总线，所有的功能模块通过 CORBA 进行交互：既可以通过 CORBA 的 IIOP 进行对象服务的调用，也可以通过 CORBA 提供的事件机制型进行互操作。利用 CORBA 作为软总线，不但能够实现分布式计算，而且能够实现系统功能的平滑扩

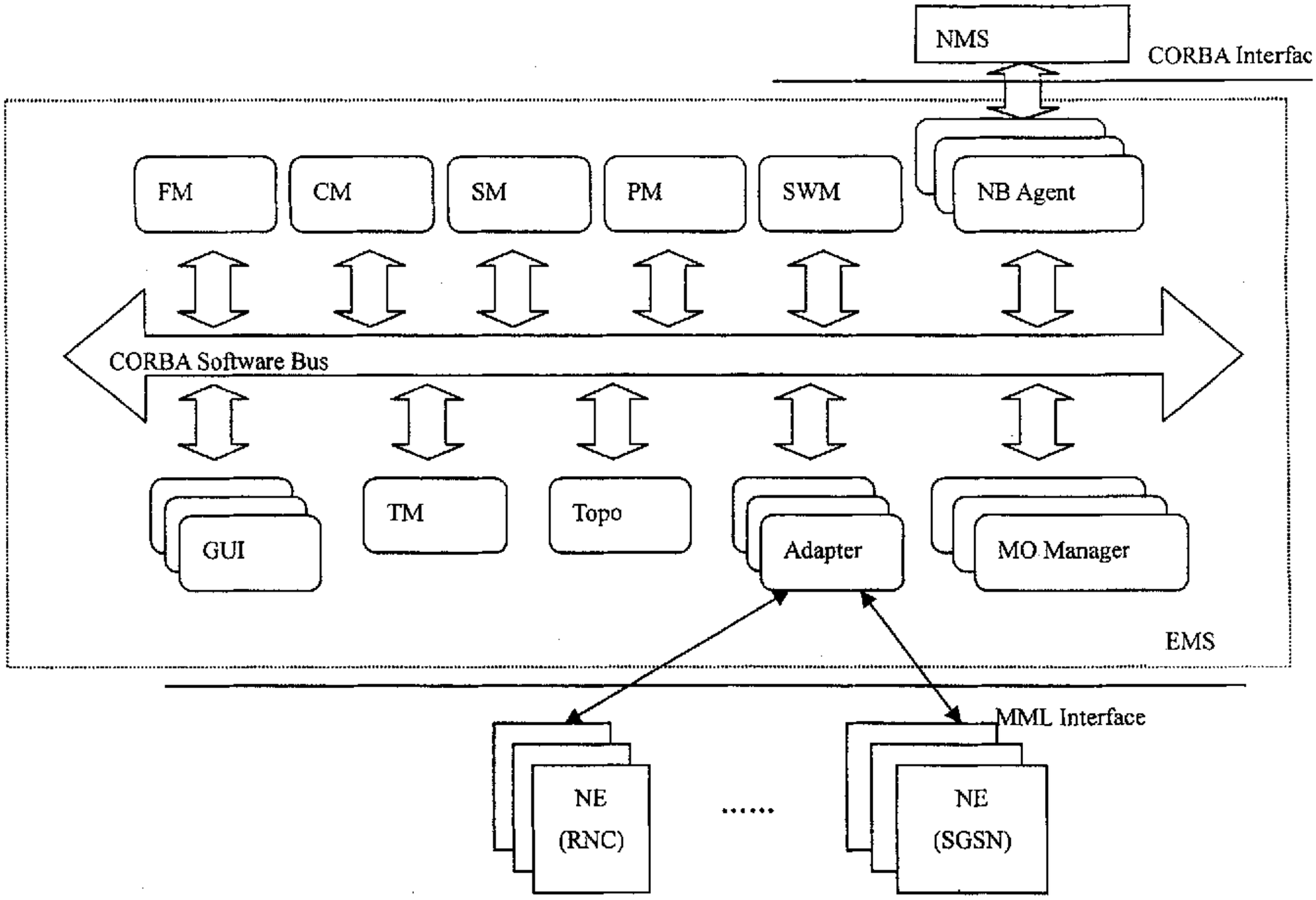



图 4.9 EMS 系统结构图

容。

系统采用 GUI/Server/Adapter 三层结构，Server 处理业务相关功能，Adapter 实现网元接入的功能：Server 层包括除 Adapter 和 GUI 外所有的模块；Adapter 模块专门来处理网元接入的问题，所有的 Manager 都在 Adapter 中实现。OSI 中定义的功能模块通过 Server 和 Adapter 协同实现。Server 与 Adapter 的分割使得 Server 层彻底屏蔽了网元相关性，使其能够专注于业务功能的开发实现。

系统通过 MO Manager 来管理 MIT：MO Manager 通过 Adapter 从网元获取相关的 MIB 信息，并写入 MIT；同时为其他模块提供对象查询接口；MIT 与 MIB 对象信息的一致性通过事件机制来保证。

系统通过 NB Agent 来实现对上层网管标准接口的提供，为了满足不同上层网管的需求，可以通过 NB Agent 进行适配开发：既不影响已有的系统实现，又满足了新上层网管的需求。在提供标准 CORBA 接口的前提下，还能灵活的为不同需求的上层网管提供 CORBA 接口。

系统的网元接入采用 4.5.2.4 节讨论的 Manager/Agent 结构，实现网元的动态接入，屏蔽不同子网、不同网元带来的网元管理的差异性。

系统的实现基于大量成熟的中间件产品，能够方便的实现跨平台、跨数据库，满足开发的需求，提高开发的效率。

系统实现集中的配置管理、告警管理、性能管理、软件管理等：方便管理员对整个网络进行统一管理；使得网络一致性检查成为可能；集中的告警管理使得告警的定位更精确、更智能化；集中的性能管理方便用户收集不同网元的性能数据，并作出分析；集中的软件管理极大的方便了网元的升级。

系统在对象建模、网元接入、数据传输等方面大量使用了 XML 技术，提高了系统的可重用性、灵活性，在下一章我们将会对 XML 在网元接入中的应用做讨论。

EMS 的系统框架和实现满足了 3G 移动网管系统对网元管理层的需求。

# 第五章 网元软件管理功能

## 5.1 软件管理功能

这里的软件，是一个相对于硬件的概念。传统的网元管理系统往往关注的是网元的各种硬件资源的管理，如对链路，板卡的配置、故障等。对于网元所处的软环境，即各种软件的维护，文件的存取等，往往缺乏必要的关注。软件管理功能就是鉴于这样的情形，提出的从系统全局的角度来对通信网络中各个网元节点上一些重要的文件进行统一的管理维护。主要的操作包括各种文件单元在 GUI，网管服务器，网元之间的传输，以及网元管理系统（服务器或 GUI）对这些传送内容的处理。

在前文中，我们提到标准文献中对公共管理模块的叙述中，公共管理模块提供的一个基本功能就是文件传输功能[37]。在通常的网管实现中，通过这个文件传输子模块，为网管的其他管理应用模块提供文件传输服务，支持其它管理应用功能的实现。文件传输模块在实现文件传输的操作中，不对文件内容进行任何鉴别，来自任何模块的内容都一视同仁，使用相同的方式进行传输。传输文件的来源和目的也都由使用传输功能的管理应用模块指定。公共管理模块提供的文件传输功能完全是一个被动的响应过程。

随着通信网络规模的扩大，系统活动中涉及的文件操作日益频繁，有些活动比如备份，日益成为一个常规的定时的任务，以一种自发执行的方式取代人为执行方式，能够减少系统管理操作，方便用户的使用。

另一方面，用户对网元管理功能的要求也在不断提高，对于网元设备上的文件，尤其是一些与网元管理功能密切相关的文件，用户希望网管系统中不仅能够对这些文件进行存取、传输，还要求网管系统能够直接使用这些文件，把一些在系统外进行转化、集成的工作直接移到系统内动态地完成。

在本文所述的软件管理中，我们把软件管理分成如下的几个部分：软件的管理，数据文件管理，备份恢复管理，网元适配。

### 5.1.1 软件的管理

在网络管理中，各种网元本身其实就是各种大小不一的计算机系统，支持这个系统运作的是网元的软件。和硬件一样，软件也有升级维护的需求。为此网元的管理也包括对网元软件的管理。

软件往往是一系列文件的集合，包括一些数据文件，可执行文件，甚至是一个层次目录结构。随着应用场景的不同，我们使用软件的方式也不一样。

软件的管理首先包括软件的上传，下载。传送发生在前台，服务器，网元三者之间，我们以传送的目的地是否是服务器来判定操作是上传还是下载。具体可以分为前台上传软件到服务器，前台从服务器下载软件，服务器下载软件到网元，网元上传软件到服务器。在网管前台和网元之间没有直接的连接，所以相互之间也没有文件的传输。

除了对软件的传送操作，当软件位于网元时，我们可以通过前台控制网管服务器对网元上的软件进行操作。这样的操作包括激活（Activate），去激活（Deativate），装载（Load），卸载（Unload）功能。

装载，即当我们将一个网元所运行的软件传输到一个网元上时，网元是将它作为一个普通的文件看待的。通常，网元本身也是一个大的系统，有自己的文件体系。一个软件被传送到网元时也是被作为一个普通的文件看待的。我们将软件传到网元的目的是为了使用它，要做到这一点，我们首先是将软件安装到网元的适当位置，比如某个基架的某块单板。

激活，软件被转载了并不表示软件处于运行的状态，出于可靠性的考虑，网元在部件上

往往有一些冗余性的设计，比如多种单板实现同样的功能相互备份。在装载时我们往往将软件装载到每个单板，但通过一个激活命令，我们可以选择一个具体位置上的软件，开始它的真正运行。

去激活，表示停止一个处于激活状态的软件运行。

卸载，类似于我们在操作系统中卸载某个应用程序，我们通过一个命令将网元上被装载的软件或补丁卸载掉。

另外在软件中还有一种特殊的类型，补丁。补丁与软件一样具有上述的各种传输操作和命令操作。但由于补丁并不能独立的运行，而是依赖于某个软件，所以在补丁的命令操作中又增加了一个确认操作，用来确认网元上的补丁，是否能够进行装载，激活操作。

### 5.1.2 数据文件管理

除了要了解和控制网元上软件的一些运行情况，有时我们也要通过对一些数据文件的分析来判断网元的运行状况，或者记录网元的操作日志。这时我们需要通过在网元和网管系统之间进行数据的交互。

软件管理只提供了数据的上传，下载功能（前台与服务器，网元与服务器之间）。对于网管系统来说，数据只是一个文件的集合，没有特殊的意义。具体的意义和使用都是由网元指定的，不在本文的讨论范围之列。

### 5.1.3 备份恢复管理

一个网管系统包含各种各样的数据，不仅是网管服务器本身为实现管理功能所必须的数据，也包括各种网元的数据。这些数据有些直接影响到移动网络提供服务的质量，甚至服务的可用性，所以实现文件的备份就显得尤为必要。

文件备份包括两个方面，网管服务器文件的备份，网元文件的备份。对于网元文件的备份，可以采取两种方式处理。一种是对网元数据在服务器上进行集中备份，为此，我们必须发送指令在网元侧首先进行备份的操作，完成之后，将这些数据传到网管服务器进行备份。另外一种，我们可以通过发送指令到网元，直接由网元在自己的本地系统中备份。

### 5.1.4 网元适配

网元管理系统出于不断的发展变化中，各大设备厂商不断推出新功能或者对一些已有功能进行扩展以应对用户日益提高的要求。所以网元设备的版本不断推陈出新。网元的版本更新带来频繁的网管接口变化，通过配置文件的方式定义网管操作接口，同时提供一套机制动态地更新配置文件，对于提高用户对网管产品的满意程度有着极大的意义。

常见的配置文件包括命令描述文件和联机帮助文件。命令描述文件提供了网元支持的命令接口，网管服务器以此为基准向各个网元发送操作命令。联机帮助文件提供了对网元的故障，命令行，性能指标等项的说明。

根据对这些配置文件使用方式的不同，有时还要对这些配置文件进行一些特别的处理。以帮助文件为例。由于各个网元对故障采用相同的编号方式，对于相同编号的故障，在读取其联机帮助信息时，就不能正确的选取是哪种类型的网元所产生的故障。所以在将这些网元的联机帮助配置文件合入到集中网管系统中时，我们就要根据网元的类型，以一种类似于编程语言的名空间的方式，为每种网元类型的每个联机帮助项分配一个系统唯一的标志，通过这个标志访问联机帮助。

## 5.2 文件传输机制

文件管理涉及大量的文件传输操作，同时也存在许多 CORBA 调用，如何将两种协议整合在软件管理服务之中，是一个关键问题。OMG 提出了一种将文件传输操作封装到 CORBA 调用中的标准，即 FTAM/FTP-CORBA。这个接口的作用是不管网络不见采用什么样的协议，

OSS 都可以对网络部件以相同的方式进行文件操作。它的优点主要有：

OSS 可以用与平台无关的方式配置，而且可以采用任何一种编程语言实现。此外有了该接口，电信运营者还可以设置轻量级的 OSS，通过 IDL 接口来专门管理这些网络部件。

网络底层的传输协议被专用接口屏蔽了，这样 OSS 就无需知道网络部件采用的 FTP 协议还是 FTAM 协议。

如果需要添加新的处理和管理文件的方法，在使用该接口的时候也比传统方式更容易一些，此时只需要在接口上继承或者直接添加新的方法就可以了。

把一个大型的 OSS 迁移到别的地方的工作变得更加容易了，这是因为采用了同样的 IDL 接口，文件管理部分只需要少许改动或者不要做改动就可以使用。

另外，由于采用了 CORBA 接口，所有数据都在 ORB 总线上传输，网络管理系统可以充分利用 CORBA 本身的优势。

FTAM/FTP-CORBA 接口由四个部分组成：

(1) 虚拟文件系统（Virtual File Systems），由 VirtualFileSystem 接口指定。VirtualFileSystem 接口为用户提供了一个使用特定文件系统的标准接口。在一个用户鉴权成功以后，用户就和 VirtualFileSystem 建立了可信任的连接。对于每一次成功的鉴权，VirtualFileSystem 都会为这个用户产生一个 FileTransferSession 对象，用户可以通过这个对象对文件系统进行操作。用 idl 接口表示为：

FileTransferSession login(in IString username, in IString password, in IString Account, out Directory root) raises(SecurityException)

(2) 文件传输对话（File Transfer Session），由 FileTransferSession 接口指定。FileTransferSession 必须专门为用户和远程实际文件服务器保持一个连接。此外，它还必须保证用户的操作正确地映射到实际文件服务器中去。对应的，FileTransferSesion 也必须把文件服务器的返回消息正确反映给客户，返回消息应该是和协议无关的。FileTransferSession 接口利用了 AVStreams（AVStreams 接口是 OMG 为 CORBA 环境下的数据传呼而定义的接口）的服务来传送文件或者文件的集合。在文件传输的时候，FileTransferSession 必须同时管理和调用 AVStreams::StreamCtrl 和 AVStreams::StreamEndPoint 接口对象。在这之前，FileTransferSession 必须首先和对应 FileTransferSession 绑定。

void bind(in FileTransferSession peer)raises(AVStream::streamOpFailed);

void unbind(in FileTransferSession peer)raises(AVStream::streamOpFailed);

FileTransferSession 的绑定和解除邦定的操作都会带动 AVStreams:: StreamCtrl 的绑定与解除绑定的自动执行。在 FileTransferSession 绑定之后，就可以利用如下的方法来实现文件传输操作了：

void transfer(in File src, in File dest)raises(SecurityException, VFSException);

void insert(in File src, in File dest, in long offset)raises(SecurityException, VFSException);

void append(in File src, in File dest) raises(SecurityException, VFSException);

Directory createDirectory(in FileName name) SecurityException, VFSException);

void delete(in File file)raises(SecurityException, VFSException);

File createFile(in FileName name)raises(SecurityException, VFSException);

void logout();

(3) 文件接口提供了到文件传输服务器上物理的文件的代理。它提供了基本的文件服务与操作。

(4) 目录接口则是一个特别的文件接口，它管理一组与之相关的文件。它不仅继承了来自文件接口的属性，还提供了引用它所关联的一组文件的方法。

Interface Directory : File {

```
    typedef sequence<File> FileList;
    void list(in unsigned long howMany, out FileList[], out FileIterator fi);
};
```

这样的方式，可以将 FTP 屏蔽到 CORBA 中，从而实现了在网管系统的客户端、服务器，以及网元之间文件传输接口的统一，但是却有着较高的实现代价。在软件管理中，客户端、网元对 FTP 的使用都是相对简单的，它们都只是充当一个 FTP 客户端的角色，从网管系统的 FTP 服务器下载软件、数据或者上传软件、数据到服务器上。在网管系统的客户端或者网元侧实现一个虚拟文件系统的代价比较高。

但是 FTAM/FTP-CORBA 的这种虚拟文件系统的思想还是在软件管理系统的服务器端得到了很好的体现和扩展。在软件管理中，我们也建立了这样一个类似的虚拟文件系统。这个系统中的单元实体并不是文件，而是一些由文件组成的软件管理对象，比如软件、数据，它们往往是由多个文件构成，并有一定的识别机制来确保结构的完整性。

## 5.3 软件管理流程分析

通过对上述功能的分析可以发现，软件管理的主要功能可以概括为如下的几种：

### 5.3.1 软件管理对象在客户端和服务器之间的传输

在软件管理中存在三种通讯接口，即 ftp 文件传输接口，CORBA 调用接口，以及网元与网管系统之间的交互协议 MML 接口。

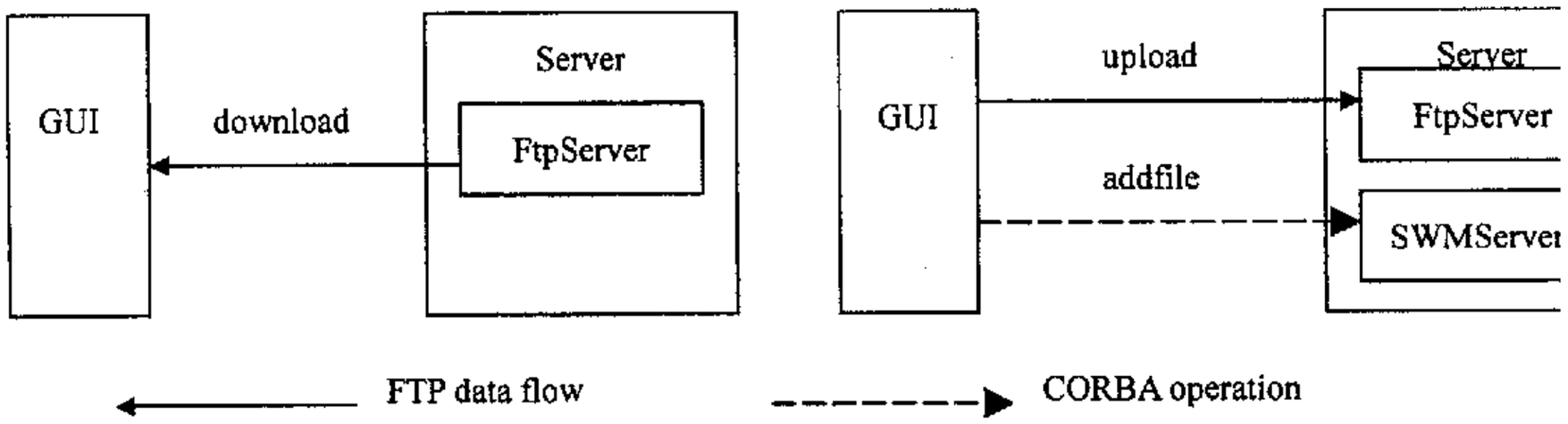



图 5.1 GUI 与 Server 间的上传、下载流程

图 5.1 显示了 GUI 和服务器之间的上传、下载流程，在这个过程中不涉及网元。当 GUI 下载一个软件或数据时，完全是一个 ftp 操作。而当 GUI 上传软件或数据时，除了进行 ftp 操作，还要通过一个 CORBA 调用，通知服务器将所上传的软件或数据的信息登记入库，纳入管理体系。

另外，由于软件是一个具有严格定义的文件集合，所以在软件的上传之前要经过一个验证工作。这个验证工作是通过对一个名为 vercfg.xml 的文件的解析来进行的。下面给出了一个 vercfg.xml 文件的例子。

```
<?xml version="1.0", encoding="iso-8859-1"?>
<Release xmlns=...>
    <File isM2000Readabel="true">
        <TypeName>NodeBSW</TypeName>
        <TypeCode>0000</TypeCode>
        <FilePath>NodeB.pck</FilePath>
        <SoftwareVersion>BTS3802CV100R003C02B072</SoftwareVersion>
        <CRCValue>1234</CRCValue>
```

```
    <FileLength>5310559</FileLength>
    <TypeDescription>NodeBSoftware</TypeDescription>
  </File>
  <File isM2000Readabel="true">
    <TypeName>NodeBRom</TypeName>
    <TypeCode>0001</TypeCode>
    <FilePath>NBRom.pck</FilePath>
    <SoftwareVersion>BTS3802CV100R003C02B072</SoftwareVersion>
    <CRCValue>1234</CRCValue>
    <FileLength>2147398</FileLength>
    <TypeDescription>NodeBBootrom</TypeDescription>
  </File>
</Release>
```

根元素"Release"表示这是一个软件版本，其中包括两个文件(由 File 元素指定)NodeB.pck 和 NBRom.pck，其类型分别是 NodeBSW 和 NodeBRom（NodeB 表明了软件所属的网元类型是基站类型，SW，Rom 是两种软件类型，另外还有一种补丁软件）。

当 GUI 上传软件时，选中一个目录，系统首先会搜索 vercfg.xml 文件，然后根据 vercfg.xml 中定义的文件列表逐个搜索对应的文件，并验证文件的属性是否与 vercfg.xml 中定义的一致。只有满足了所有的条件才被认为是有效的软件，才会进行后续上传、入库的操作。

### 5.3.2 软件管理对象在网元和服务器之间的传输

服务器与网元之间的软件，数据的传输同样是基于 FTP 协议的，但是由于传输命令是在 GUI 发起的，而传输的双方却是服务器与网元，所以传输过程是被间接执行的。GUI 首先发起一个 CORBA 调用，通知服务器发起与网元之间的传输操作。服务器通过一个 NEEngine 向网元发送 MML 命令，由网元来执行 FTP 操作，向服务器上传或从服务器下载相应的文件。服务器接收网元的状态事件，更新对网元状态信息的记录，并将状态变化反馈到 GUI，刷新界面的显示。

流程如图 5.2 所示。

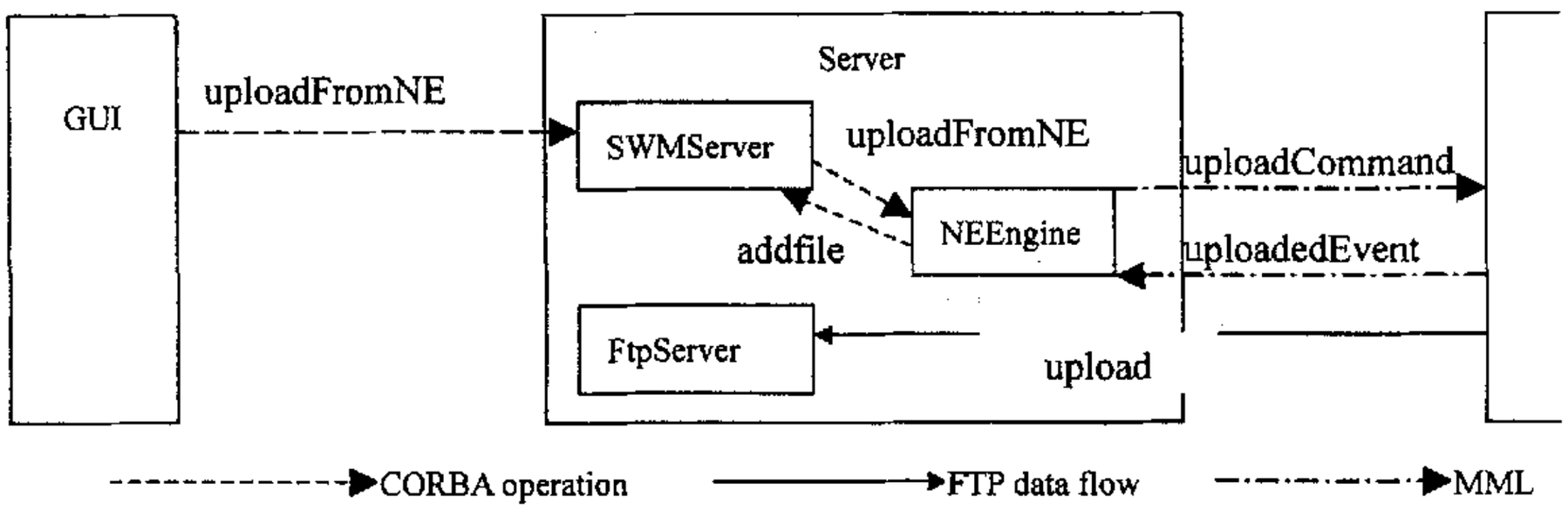



图 5.2 Server 与网元间的上传流程

图 5.2 显示了一个从网元上传数据的流程，下载软件、数据的过程与此类似，除了下载并不需要调用 addfile 这样的 CORBA 接口来更新数据库。

### 5.3.3 网管系统对网元的软件管理命令操作

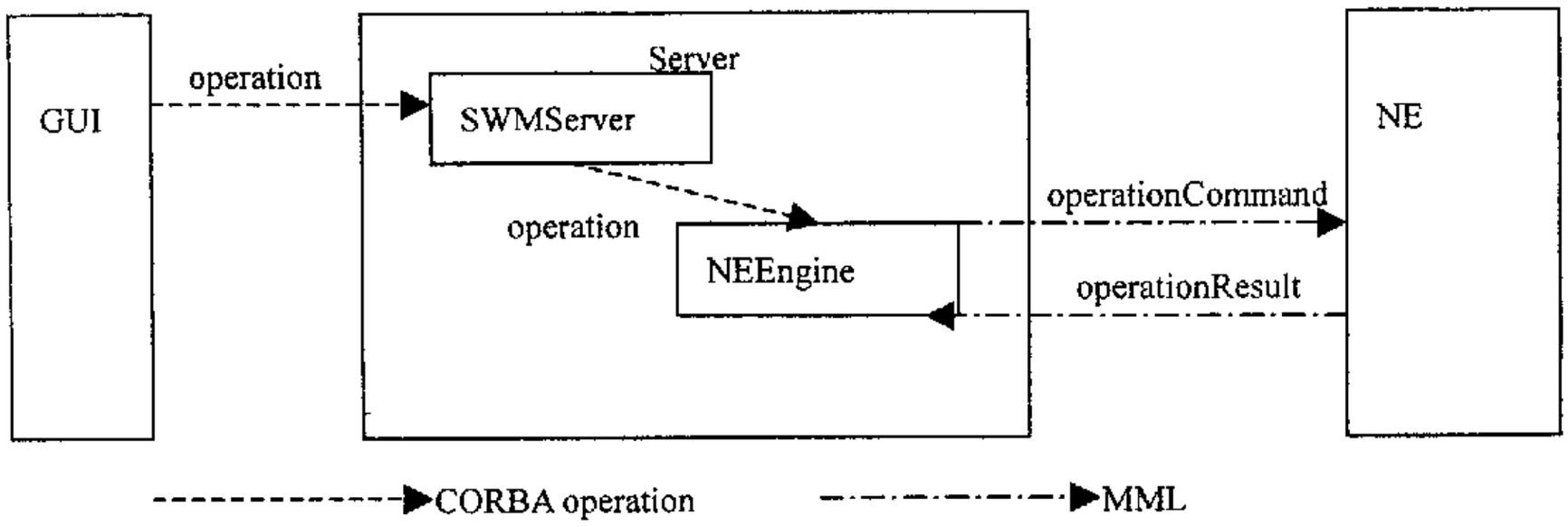



图 5.3 网元命令操作流程

网管系统对网元的操作，首先是由 GUI 发起一个 CORBA 操作，再由软件管理服务向网元发送 MML 命令报文来实现。软件管理服务接收网元的事件，更新状态，并将状态变化反馈到 GUI。

### 5.3.4 网元适配流程

网元适配的流程其实是一个两步的过程。首先是网元与服务器之间进行 FTP 操作，服务器更新自己的文件信息树。当 GUI 在使用某个配置项时，发现本地没有对应版本的配置项，就向服务器进行查询，服务器检查维护的一个文件信息树，找到可用的更新就向 GUI 返回所需配置项的路径。GUI 通过 FTP 操作获取所需的文件到本地，再进行所需的处理。

当前两个主要的配置项，是网元的 MML 命令行配置文件和网元的联机帮助文件。命令行配置文件的更新不需要任何特殊的处理，而网元联机帮助文件的使用需要有一个合入的过程。

在以 Java 开发的应用程序中，联机帮助是根据 JavaHelp 规范来提供的。JavaHelp 规范定义了类似于 CHM 的一套联机帮助系统。通过定义了几个头文件，来整合所有的 Html 编写的帮助页面。几个重要的头文件包括 help.hs, help_Toc.xml, help_map.jhm, 它们都是 xml 格式的文件。help.hs 的作用是指定整个帮助的一些属性如标题和其他的头文件，即 help_Toc.xml, help_map.jhm 文件。help_Toc.xml 文件是一个内容表(Table Of Content)文件，浏览帮助时，左窗口所显示的树形目录结构就是通过 help_Toc.xml 文件指定的。其 DTD 文件定义如下：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!ELEMENT toc (tocitem*)>
  <!ATTLIST toc version CDATA #FIXED "1.0">          //定义了 toc 文件的版本
  <!ATTLIST toc xml:lang CDATA #IMPLIED>
<!ELEMENT tocitem (#PCDATA | tocitem)*>
  <!ATTLIST tocitem xml:lang CDATA #IMPLIED>
  <!ATTLIST tocitem text CDATA #IMPLIED>             //树形目录表上显示的标题
<!ATTLIST tocitem image CDATA #IMPLIED>              //用来指定树形标题项的图标
<!ATTLIST tocitem target CDATA #IMPLIED>             //该 tocitem 项的标识字符串
```

toc 文件的根节点就是一个 toc 元素，它包含若干个 tocitem，tocitem 既可以是一个包含若干个 tocitem 子项的目录项，也可以作为目录表项上的一个叶节点，只关联一个 html 页面。一个包含子项的 tocitem 也可以通过 target 属性指定一个关联页面。当我们打开一个帮助文件时，选择某个目录表项，就可以打开与之关联的 html 页面。

tocitem 与 html 文件的关联是通过 tocitem 的 target 属性，target 的标识字符串通过 help_map.jhm 中定义的映射关系与一个 url 相关联，从而间接的实现 tocitem 与文件 url 之间的映射关系。这是选择目录表项就能显示一个 html 文件内容的基础。

help_map.jhm 文件的 DTD 定义如下：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!ELEMENT map (mapID*)>
<!ATTLIST map version CDATA #FIXED "1.0">          //定义了 map 文件的版本
<!ATTLIST map xml:lang CDATA #IMPLIED>
<!ELEMENT mapID EMPTY>
<!ATTLIST mapID xml:lang CDATA #IMPLIED>
<!ATTLIST mapID target CDATA #REQUIRED>            //字符串定义的文件标识
<!ATTLIST mapID url CDATA #REQUIRED>               //文件的 url 路径
```

JavaHelp 在它的帮助体系中引入了一个 target 来标识某个文件，并提供了通过这种标志在程序中直接访问某想帮助页的机制。由于字符串的可定制性更好，这为帮助的使用提供了更大的灵活性，这一点在综合网元管理系统中起了很大的作用。

综合网元管理系统集中了网元本地维护的所有功能，具备了执行各种网元的操作命令的能力，但是对于各种网元累加起来的数千种指令，以及每种指令多达十几个的参数，没有对应的联机操作手册是无法想象的。另一方面，随着网元种类的增加，网管系统的故障模块所要处理的故障种类也不断增加，在故障处理中，我们也需要联机手册来帮助我们理解各种网元故障的含义，从而做出相应的处理。

在通过 FTP 取得网元的帮助后，我们需要把网元的帮助合入到网管系统的帮助中，以便于网管系统使用。具体的合并一方面是把网元帮助文件拷贝到网管系统帮助的某个子目录下面，另一方面是 help_Toc.xml, help_map.jhm 文件的合入。这时遇到的一个问题就是 target 字符串所代表的标识符的名称的冲突问题。网元的联机帮助系统一般是采用 CHM 进行开发的，CHM 没有字符串到 url 的这种映射关系，而是直接将目录项与 url 关联。在从 CHM 到 JavaHelp 的转换中，采用了一个简单的将 url 通过某种转换形成一个 target 字符串的处理。比如一个相对路径为 "help/mml/lst software.htm 的 html 文件生成了一个"help.mml.lst_software"。当这样的标识只在一种网元中使用时，是没有问题的，因为每个帮助页面的名称都是对应与命令名称或是告警号的。但当网元帮助应用于整个网管系统时，来自另外一个网元类型的帮助也有可能具有相同的目录结构，相同的命令名称，尤其是随着接口日趋规范统一，这种现象非常普遍。所以，在合成网元管理系统的帮助时，要解决类似于编程语言中的名空间冲突的问题。

为此，我们必须对网元帮助的 target 进行定制，在合入网元帮助时按照某种规则对其进行转换，保证其系统唯一性。在转换过程中，target 标识还有如下的要求：必须是明确有意义的，能够被应用程序根据某种规则构造出来。为了满足这一要求我们定义了如下的 target 规则：

网元帮助 target ::= 网元类型名称 +"_"+ 网元版本号 +"_"+ 帮助类型 +"_"+ 特征字串

网元类型名称，用来区分帮助所应用的网元类型，这是一些在系统内被普遍使用的常量。

网元版本号是一个只含有字符、数字的字符串。

帮助类型是一个枚举的字符串常量，当前定义了三种类型"MML"，"Alarm"，"Performance"。

特征字串根据帮助类型的不同，也有不同的格式。对于 MML 帮助，特征字串是一个形如"LST SOFTWARE"这样的 MML 命令名称，对于告警，则是一个表示告警号的数字。

帮助的合入是网元适配的一个主要方面，在帮助合入的过程中，通过上述的这种 target 的转化，使网元的联机手册能够直接在程序中被应用，大大提高了系统操作的易用性。

# 第六章 软件管理模块设计与实现

软件管理系统的实现由主要由三个部分构成，即服务子系统，适配子系统，GUI 子系统。由于软件管理涉及大量的文件传输操作，在服务子系统中集成了一个 ftp 服务器。它们的交互关系如图 6.1 所示。

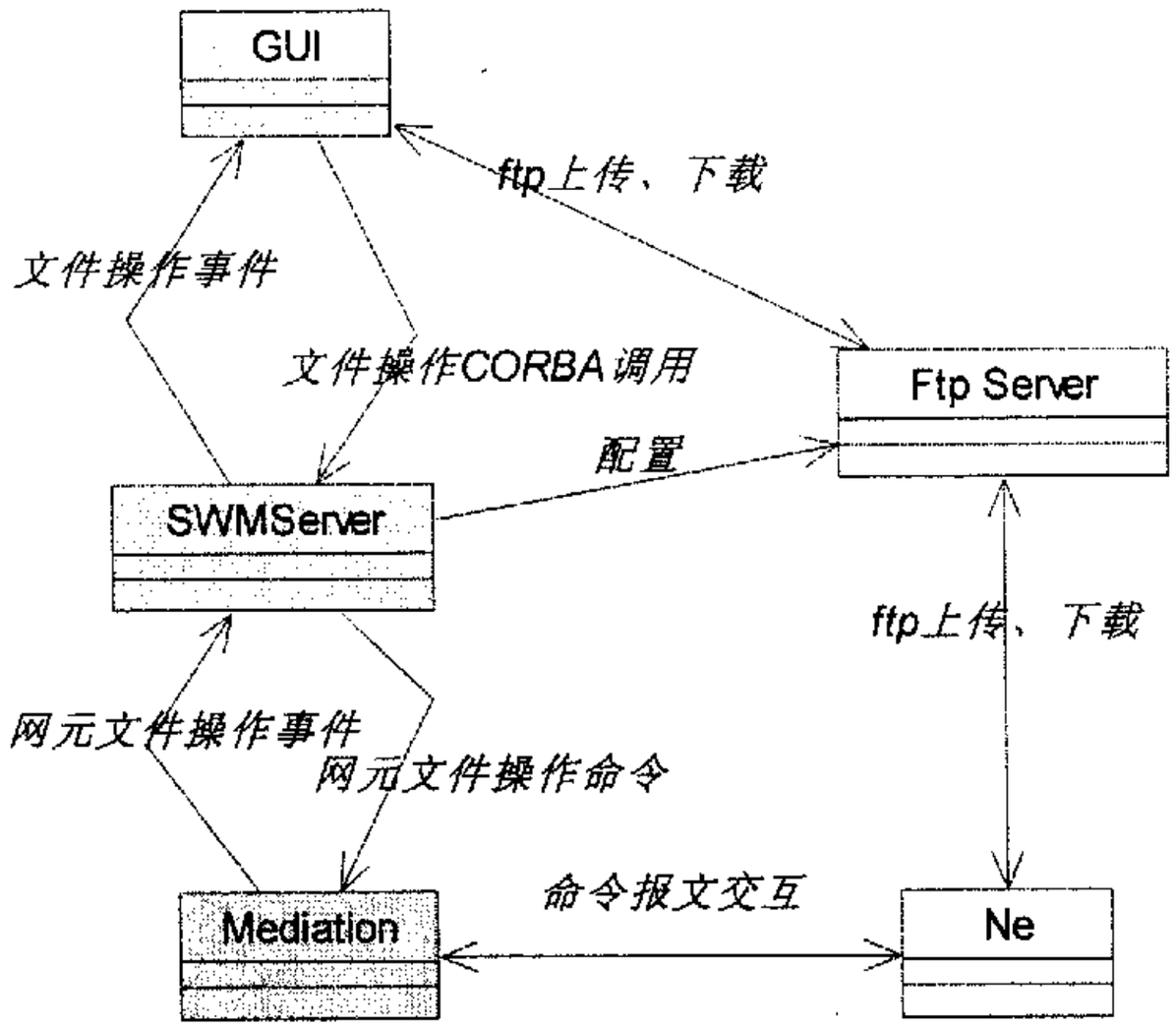



图 6.1 软件管理子系统交互关系

下面定义了一些枚举型的常量，这些常量在系统中被多次用到。

enum ElementType{Software, SoftDir, Data, DataDir, Patch, PatchDir, License, LicenseDir, Other, OtherDir, HelpDir, CPFType, DummyElementType};

ElementType 描述了系统中软件管理所引用的各种文件资源的类型，在前面的描述中，我们概括为软件，数据两类。这种分类的依据是软件具有传输、执行两种特征，而数据只具有传输特征。

enum NEOPTYPE{NEopDownlaod, NEopUpload, NEopActive, NEopLoad, NEopSYN, NeopUnknow};

NEOPTYPE 定义了软件管理中操作网元的类型。

enum FtpOpType{DOWNLOAD, UPLOAD, OtherOPType};

FTP 操作的类型，定义了下载，上传和其他三种。增加了其他是为了保证对一些可能的特殊应用的支持。

enum FTPOrigin{GUI, NE, HOST, OtherOrigin};

描述了 Ftp 文件的来源，可能是 GUI，网元，服务器或者其他。这里其他也是出于一种可扩展性的考虑而设置的。

## 6.1 服务子系统设计与实现

服务子系统是一个建立在网管平台上的 CORBA 服务，使用 C++语言开发，同时运用了 ACE 这样的中间件技术来屏蔽了底层与系统相关的细节。软件管理服务子系统的类结构

图如图 6.2 所示。

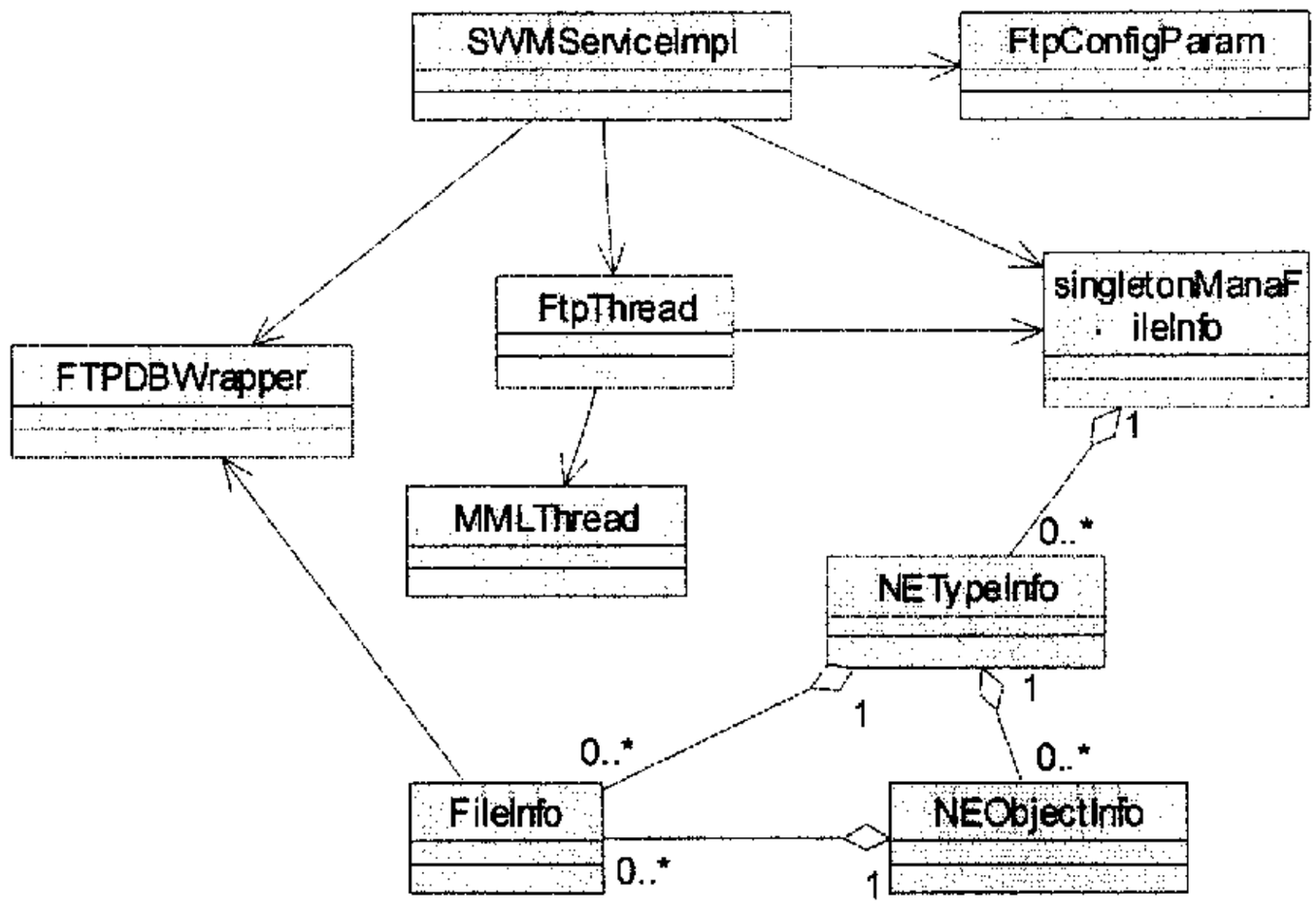



图 6.2 软件管理服务子系统类图

最主要的几个类包括 SWMServiceImpl, SingleManaFileInfo, FtpThread。

SWMServiceImpl 实现总体的功能，是软件管理服务的主对象，在网管系统运行时作为一个服务进程在系统中运行。它是 GUI、其他管理应用服务调用软件管理服务的入口。

SWMServiceImpl 的主要作用包括初始化 Ftp 服务器的参数设置，建立一个软件管理信息树来维护对软件管理中被管理的各种软件、数据的访问，启动一个 FTPThread 线程来进行各种软件传输、操作任务的状态更新。

SWMServiceImpl 的初始化过程表示如下：

```
initialize()
{
    判断是否已经初始化;
    判断输入参数指针是否合法;
    生成 FTPConfigParam 类对象，进行检查;
    初始化本地化资源文件;
    记录传入参数;
    复制 POA 和 ORB;
    生成 Oid;
    激活进程;
    注册 naming service;
    注册 valuetype factory;
    生成 singletonManaFileInfo 对象;
    进行 singletonManaFileInfo 的初始化;
    生成 FTPThread 类对象;
    激活 FTPThread 线程;
}
```

SWMServiceImpl 包含了所有软件管理的各种操作接口，如 activeSoftware，

deactiveSoftware，loadSoftware 等操作命令接口以及 transferSoftware，transferData 等文件传输接口，对于这些接口的实现，SWMServiceImpl 的实现大致相同，都是对操作对象设置一个状态，由 FTPThread 在线程里进行状态更新。

我们以 transferData 的执行过程来说明 SWMServiceImpl 实现各种操作的过程。

```
transferData()
{
        if（输入参数无效）{ 异常退出；}
        if（没有操作权限）{ 异常退出；}
        获取 singletonManaFileInfo 对象，加读写锁；
        循环查找网元列表判断网元状态，对于每一个网元
        if 网元没有进行软件激活或文件传输任务，当前状态可用
            清除当前网元传输任务列表；
            添加当前传输任务到网元列表；
            置网元状态；
        else
            添加网元到失败网元列表；
        释放锁；
}
```

其中 SingleManaFileInfo 是一个包含了所有处于传输状态的文件信息以及被纳入管理体系的已传输文件信息的结构。他通过一个 NETypeInfo 列表来维护这些信息。

NETypeInfo 类的成员变量如下所列：

```
Class NETypeInfo {
string m_netype;                          //网元类型名称，这种网元的唯一标志。
string m_softPath;                        //软件存放路径
List<NEObjectInfo> m_NEObjectList;        //该网元类型的所有网元的软件管理信息列表
List<FileInfo> m_FileList;                //该网元类型的软件列表
}
```

软件是作为具有同种类型的网元所共有的文件数据来存放的，所以在网元类型信息中包含一个用于存放网元软件的路径信息，获取某个软件可以通过 m_softPath 与 m_FileList 中的软件名称的指定共同获得。

NEObjectInfo 类的成员变量如下所示：

```
Class NEObjectInfo{
string netype;              //网元类型
string nename;              //网元名称
string nefdn;               //该网元 fdn，用来在系统内唯一的标识一个网元
boolean  m_ftpFlag;         //该网元 ftp 标记，说明 ftp 传输的状态，上传到网管服务器
还是下载到网元
    short m_Activeprogress；  //该网元软件激活的进度
    string m_datapath;        //该网元存放数据文件的路径
    string m_licensePath;     //该网元存放许可证文件的路径
    string m_otherPath;       //该网元存放其他文件的路径
    string m_tipMessage;      //传输、命令操作过程中的一些提示消息
    int m_sessionID           //FTP 传输的 sessionid
    int m_timeCount;          //FTP 的超时计数
```

```
List<FileInfo> m_FileList;          //网管服务器保存的所有网元文件信息
Vector<FileInfo> m_FtpFileVector; //处于 FTP 传输过程中的文件信息
}
```

FileInfo 表示单个的网元数据、软件、许可证或其他软件管理对象及其传输状态等信息，它的成员变量如下所示：

```
Class FileInfo{
   string netype;                          //网元类型
   ElementType elementtype;                //对象类型，软件、数据等
   FTPOrigin m_origin;                     //软件管理的文件对象来源
   FileStatus m_fileStatus;                //软件管理的文件对象状态
   FtpOpType m_optype;                     //软件管理对文件对象的操作类型
   Time m_ftpTime                          //开始 FTP 操作的时间
   string m_neFdn;                         //软件管理的文件对象所属的网元 fdn
   long m_filesize;                        //文件对象大小
   long m_filecrc;                         //crc 校验值
   string m_patchversion;                  //补丁版本，当文件对象是一个补丁时使用
   string m_fileversion;                   //数据版本，当文件对象是一个数据时使用
   short m_fileFtpProgress;                //文件传输进度
   FtpStatus m_ftpStatus;                  //文件状态
   boolean m_FileDBStatus;                 //是否已经入库
}
```

通过 SingletonManaFileInfo，NETypeInfo，NEObjectInfo，FileInfo 这样一个四层的树形结构，软件管理完成了文件管理信息对象的建模。

SingletonManaFileInfo 除了涵盖这样一个结构，还包含关于网管服务器上 ftp 服务器的信息，其他模块或 GUI、网元与软件管理服务模块通信时可以从此处获取 ftp 服务器的信息，然后完成文件的传输操作。

当系统启动时，FTPThread 作为软件管理服务的一个线程开始执行，它的主要作用就是循环获取 singletonManaFileInfo 中各个网元、文件的状态，并对它们进行更新。

图 6.3 和图 6.4 分别显示了网元和文件的状态迁移图。

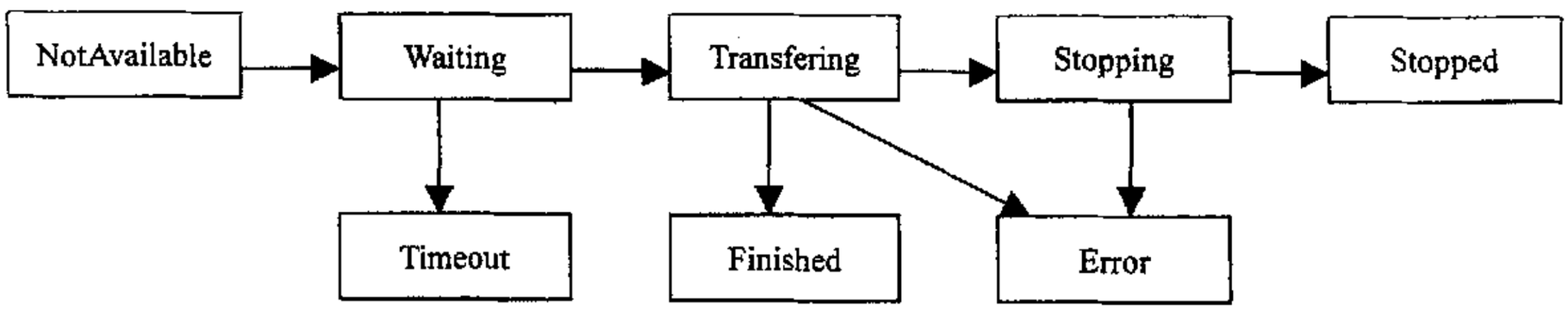



图 6.3 网元传输状态迁移图

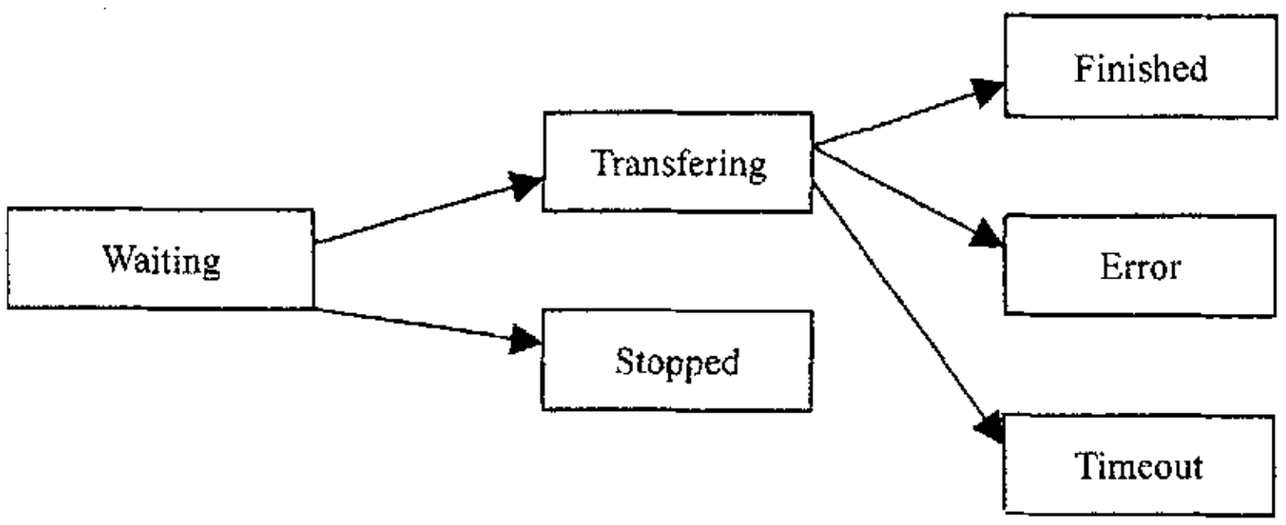



图 6.4 文件传输状态迁移图

FTPThread 实现网元和文件的状态迁移是通过 svc()方法实现的。svc()方法的主要作用就是实现网元，文件传输状态的迁移控制。

## 6.2 适配子系统设计与实现

在本文的集中网管系统中，系统使用粗粒度的建模方式，对管理对象的建模只在网元层，网元 MO 类的继承关系如图 6.5 所示。网元 NEEngine 的继承关系如图 6.6 所示。

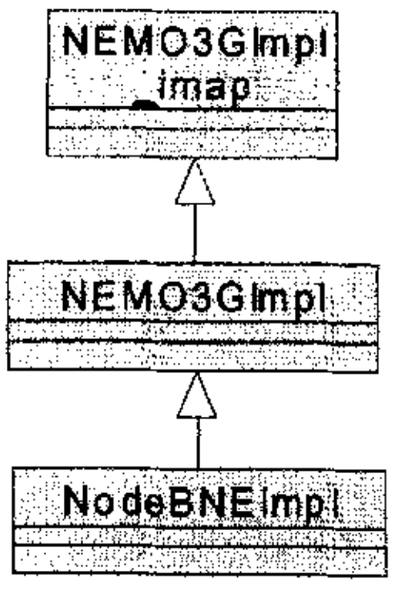



图 6.5 软件管理 MO 继承关系

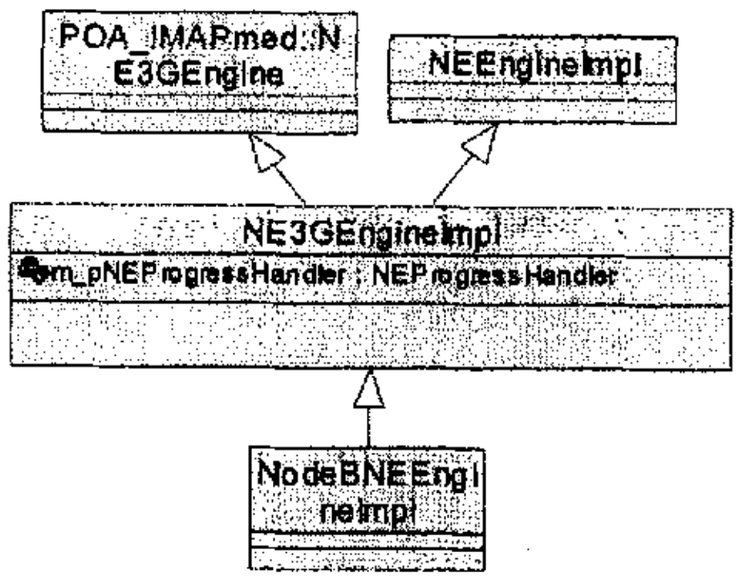



图 6.6 软件管理 NEEngine 继承关系

上面以 NodeB 为例，说明了网元 MO 和 NEEngine 的继承关系。类 NEMO3Gimpl 从网管平台的类 NEMO3Gimpl_imap 继承，在该类中实现所有软件管理操作的接口，对每个接口给出网元通用的实现，并调用 NE3GengineImpl 中相应的方法完成功能。每种网元的类从 NEMO3Gimpl 继承，如果接口实现方法和通用方法不同，需要重载接口方法。

适配子系统主要处理 Server 与网元之间的两类交互，一种就是 Server 向网元下发命令，另一种就是接收网元发送的报文进行处理。如果网元的报文和通用报文一样，不需从 NEProgressHandler 继承各自的处理类。如果网元的报文和通用报文有区别，需要生成网元各自的事件处理类，重载处理函数 processNotifyReport。事件类在 NE3GengineImpl 的初始化函数 doInit 中注册，在函数 doFini 中撤销。

## 6.3 前台子系统设计与实现

GUI 是用户使用软件管理功能的操作界面。在业界，存在两种主要的操作界面形式，应用程序界面和浏览器页面。两种方式各有优缺点，应用程序界面实现起来更加灵活，但是必须安装专用的客户端进行操作。而基于浏览器的实现方案能够在任何连接到网络上的机器上执行，但页面的表达形式往往受制于 HTML 语言本身的限制，同时在速度上也要低于应用

程序界面。在本系统中，我们选用了应用程序界面作为 GUI 的实现方案，使用 Java 语言，以便于 GUI 能够跨平台执行。

图 6.7 显示了软件管理前台子系统的类图。

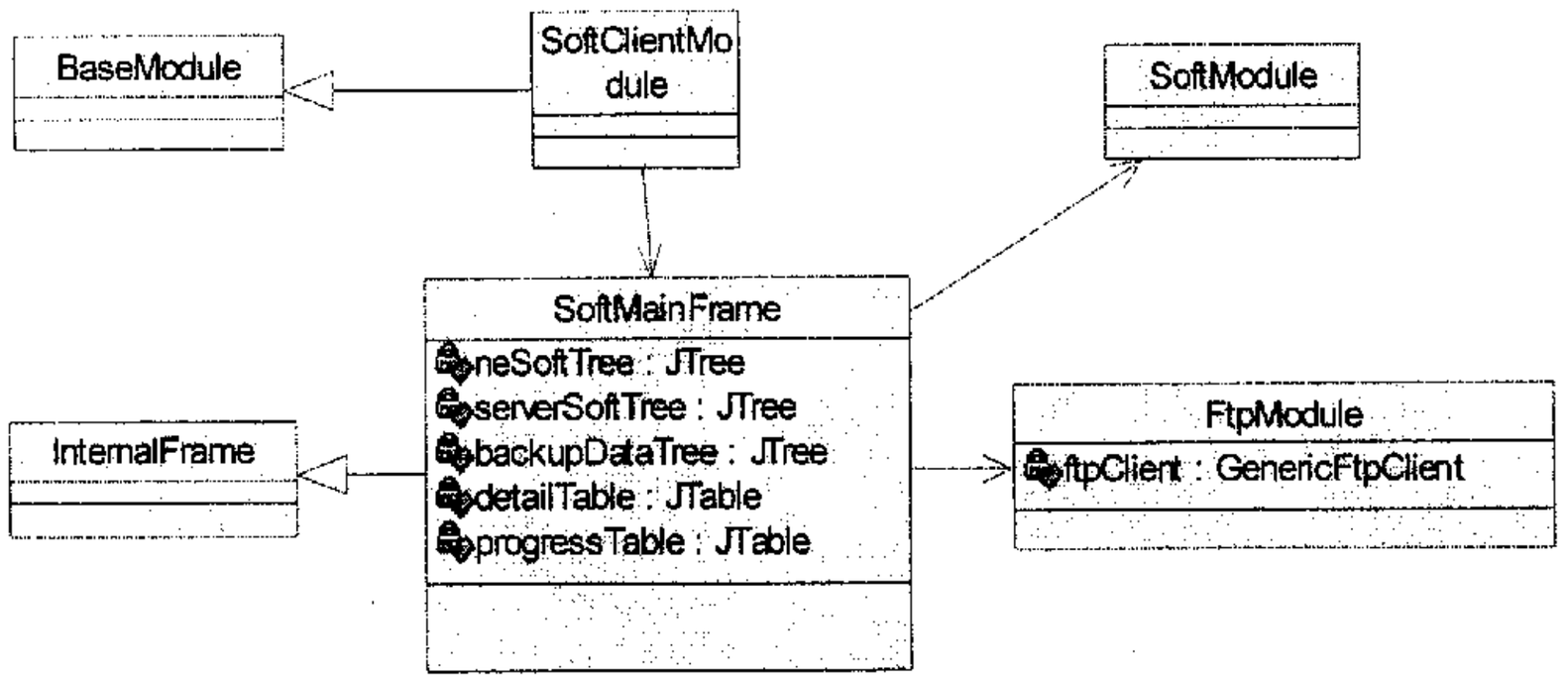



图 6.7 软件管理 GUI 子系统类图

在网管应用的前台系统的开发中，也抽取了一些公共的特性，建立了前台系统的开发框架。在网管应用的前台开发中，我们可以运用这种框架，来降低设计的复杂性，同时也提高了系统的可配置性。在软件管理的前台开发中就运用了这样的框架。

框架提供了一个 BaseModule 和一个 InternalFrame 类。我们可以通过继承 BaseModule，加入管理应用逻辑来实现自己的管理应用模块，这使得开发人员可以把开发的焦点集中到管理应用逻辑的开发上，而无需过多的关注模块的加载、关闭等外部特征。

SoftClientModule 就是这样的一个模块，通过在一个 xml 文件内指明 SoftClientModule 的全限定类名称，系统前台就可以在启动时直接加载软件管理模块。

软件管理的主窗口是 SoftMainFrame，该窗口继承自 InternalFrame，通过这种继承关系，确保了软件管理的主界面在风格上具有网管系统的整体特色。SoftMainFrame 采用左树右表显示了软件管理所管理的各种软件、数据对象。共有三种树形结构 neSoftTree，serverSoftTree，backupDataTree,分别向用户提供了对三种软件管理对象的浏览，即对网元侧软件管理对象的浏览，对 server 侧软件管理对象的浏览，以及对网元的各种备份数据的浏览。通过 tab 页可以在这几种树形结构之间进行切换。neSoftTree 按照"NE"(根节点)—网元类型名称—网元名称—文件类型—文件的层次结构显示给用户，而 serverSoftTree 按照"Server"(根节点)—网元类型名称—文件类型—网元—文件的层次结构显示给用户，backupDataTree 的结构与 neSoftTree 结构类似。对于 serverSoftTree，这种层次结构只代表了软件管理对象在软件管理服务器上的目录层次关系，sererSoftTree 上显示的每一个软件管理对象，都位于 FTP 服务器的一个层次相同的目录中，而 neSoftTree 上的每个软件管理对象是通过 CORBA 调用，从网元返回的相关信息构造而成的，网管系统中没有对应的实体。右表 detailTable 根据左树节点的选择进行动态的调整。当选中某种网元类型时，在右表的每一行显示属于该网元类型的一个网元名称，当在 neSoftTree 上选中一个文件类型的节点或在 serverSoftTree 上选中一个网元节点时，右表的每一行列出了该文件的一些详细信息。

所有的软件管理操作都是通过在三种树结构和表格上使用右键点击来触发的。在 SoftMainFrame 中通过一个 adjustPopupMenu()方法来控制右键菜单的显示，使得在选中不同的节点时可以进行不同的操作。软件管理中的两大类操作，传输操作一般通过在左树上选择相应的节点来触发，而命令操作一般通过在表格上选定对应的行来触发。

软件管理前台的一个重要功能就是可以根据网元对软件管理的支持来进行界面的定制，屏蔽一些不必要或者不支持的功能。屏蔽的方法是通过一个 xml 文件来定义在 Server 侧（即对主界面上的 serverSoftTree）以及在网元侧（主界面上的 neSoftTree）所能进行的操作。这个 xml 文件格式如下：

```
<?xml version="1.0", encoding="iso-8859-1"?>
<SWMNEConfig>
    <Server>
        <NodeB isSupport="true">                    //表示支持 NodeB 在 Server 侧的软件管理操作
            <Version isSupport="true"/>             //表示支持 Server 侧的对 NodeB 软件的上传和下载
            <Data isSupport="true"/>
            <License isSupport="true"/>
             …
        </NodeB>
        <RNC  isSupport="false">        //如果这里为 false，表示不支持 RNC 在 Server 侧的所有操作，
    //界面上 serverSoftTree 的根节点 “Server”下将不显示 RNC 这个网元类型
            <Version isSupport="true"/>
            <Data isSupport="true"/>
            <License isSupport="true"/>
             …
        </RNC>
        …
    </Server>
    <NE  isSupport="false">                     //可以在这一项配置对整个网元侧操作的支持，这一项为 false
    //那么以下的配置都将无效，界面上将不显示 neSoftTree
        <NodeB isSupport="true">
            <Version isSupport="true"/>
            <Data isSupport="true"/>
            <License isSupport="true"/>
             …
        </NodeB>
        …
    </NE>
</SWMNEConfig>
```

打开软件管理主界面时，系统首先会读取这个配置文件，根据这个配置文件生成一个表示配置信息的对象，当构造软件管理的 neSoftTree 和 serverSoftTree 时，会根据这个配置信息来调整树的结构。由于这些树是软件管理操作的入口，一些不被支持的项将被屏蔽。

# 第七章 结束语

随着通信网络的发展，网络的结构和功能的复杂性不断提高，通信网络对网络管理系统的功能要求也变得越来越严格。市场竞争的加剧，推动了开发商采用更多的开放性技术来满足不断上升的兼容性要求，并通过网管系统平台的构件来复用各种通信网管中的公共部分。在这样的要求下，基于 CORBA 的中间件技术在网管系统的开发中得到了广泛的应用。CORBA 具有平台无关性和丰富的语言映射机制，用于开发网管系统十分便利。

在用 CORBA 进行网管应用功能的开发中，面临的主要问题就是如何将一些开发商私有的网络协议和一些具有特定功能的协议进行交互，集成，发挥各种协议的优点来共同完成某些管理任务。本设计的主要创新点体现在软件管理模块的设计实现中。网元软件管理应用功能，是一项综合了文件传输，远程控制的管理应用。在以 CORBA 服务为基础的主框架中，如何实现基于 FTP 协议的传输功能和基于 MML 的远程操作功能成为了主要的挑战。本文对这一架构给出了一个完整的设计。

同时，在 TMN 标准文献的基础上，本文对软件管理功能进行了扩展和加强。这是在用户对网管应用的功能要求不断提高的基础上发展起来的，并在实际使用中得到了充分的肯定。

本网络管理系统已经在国内某知名电信公司实现了商用，从两个 3G 试验网和一个 3G 商业网上的运作状况看来，本网管系统较好地完成了各个管理功能，而且运作效率教高。

由于网元软件管理还处在不断发展完善中，某些类型的网元对软件管理的支持还不充分，软件管理对象的标准工作还在进行，所以我们还有许多工作要做。

# 参考文献

[1] ITU-T Recommendation M.3010, Principles for a Telecommunications Management Network. 1996.

[2] ITU-T Recommendation X.744.1, CORBA-based TMN software management service. 2003.

[3] ITU-T Recommendation M.3120, CORBA generic network and network element level information model. 2001.

[4] 夏海涛, 詹志强. 新一代网络管理技术. 北京邮电大学出版社. 2002.12。

[5] WCDMA 系统基本原理. 第九章 3G 网络管理。

[6] 3GPP Organizational Partners. 3GPP TS 32.401 Performance Management (PM) Concept and Requirements. 2004.

[7] 3GPP Organizational Partners. 3GPP TS 32.403 Performance Management (PM) Performance measurements - UMTS and combined UMTS/GSM. 2004.

[8] ITU-T Recommendation X.722, Information technology - Open Systems Interconnection - Structure of Management Information: Guidelines for the definition of managed objects, 1992.

[9] ITU-T Recommendation X.780, Guidelines for Defining CORBA Managed Objects. Geneva, 2001.

[10] ITU-T, Q.811, Lower Layer Protocol Profiles for the Q3 Interface.

[11] ITU-T, Q.812, Upper Layer Protocol Profiles for the Q3 Interface.

[12] 周华春, 刘峰, 李红辉 铁路电信管理网的 Q3 接口研究,北方交通大学学报,2001.4, 25(2)。

[13] 成玉荣，CORBA 与 TMN 接口（Q3）适配技术研究，无线电通信技术，2003.29(1)。

[14] 吕铭 等，CORBA 命名服务在 TMN 中的应用，计算机应用，2002.7，22(7)。

[15] 王忠，陈海清，CORBA 事件服务的应用研究，华中科技大学学报，2001.11。

[16] 亓峰，基于 CORBA 技术的网管标准及新一代网络管理问题，通信世界，2001. 26。

[17] 宋光磊，张钥，顾冠群，CORBA 网管框架研究，系统仿真学报，2001，11。

[18] 寿永艳，电信综合网络管理系统的 CORBA/SNMP 网关的研究与实现，2002.6。

[19] Douglas C.Schmidt, Stephen D.Huston. C++网络编程 卷 1: 运用 ACE 和模式消除复杂性，华中科技大学出版社，2003.12。

[20] Douglas C.Schmidt, Stephen D.Huston. C++网络编程 卷 2: 基于 ACE 和框架的系统化复用，华中科技大学出版社，2003.12。

[21] ISO 9596, Information Processing Systems - Open Systems Interconnection - Common Management Information Protocol", Geneva, 1991.

[22] ISO 10040, Information Processing Systems - Open Systems Interconnection - Systems Management Overview, Geneva, 1992.

[23] 3GPP TS 32.101, Telecommunication management: Principles and high level requirements, 2002.

[24] 3GPP TS 32.102, Telecommunication management; Architecture, 2004.

[25] 3GPP TS 32.341, File Transfer (FT) Integration Reference Point (IRP): Requirements, 2004.

[26] 3GPP TS 32.342, File Transfer (FT) Integration Reference Point (IRP): Information Service (IS).

[27] 3GPP TS 32.343, File Transfer (FT) Integration Reference Point (IRP): Common Object

Request Broker Architecture (CORBA) Solution Set(SS).

[28] 3GPP TS 32.341, File Transfer (FT) Integration Reference Point (IRP): Requirements, 2004.

[29] 3GPP TS 32.342, File Transfer (FT) Integration Reference Point (IRP): Information Service (IS).

[30] 3GPP TS 32.343, File Transfer (FT) Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set(SS).

[31] Object Management Group, CORBA-FTAM/FTP Interworking, 2002.

[32] 阚璞, CORBA 在 TMN 中应用的研究和实践，硕士学位论文，中国科学技术大学，2000。

[33] Sun Microsystems, JavaHelp System, http://www.sun.com/products/javahelp/.

[34] 龚正虎，廖丽惠，黄杰，陈琳，综合系统管理与网络管理实现技术的研究，计算机工程与应用， 2002 Vol.38 No.23.

[35] Object Management Group, CORBA-FTAM/FTP Interworking, 2002.

[36] 阚璞, CORBA 在 TMN 中应用的研究和实践, 硕士学位论文，中国科学技术大学，2000。

[37] ISO 7498-4, Information Processing Systems - Open Systems Interconnection - Basic Reference Model, Geneva, 1992.

# 致谢

论文完成之际，我要特别感谢指导老师陈抗生教授和王勇副教授，如果没有这两位老师对我的悉心指导，我就不能如此顺利地完成课题的设计。两位老师平易近人、知识渊博、为人正直，注重培养学生的动手解决问题能力，我从他们身上学到的不仅仅是知识，还有如何学习、研究的方法。

同时我要感谢浙江大学信电系的黄爱萍教授，感谢黄老师对我在外实习期间的关心和指导。

感谢我的一位良师益友 Tim Chen 博士，他不仅指导我如何解决课题中遇到的难题，更是教导我思考问题的方法。

感谢我华为的导师徐侃烈博士，他对我的教诲颇多，教会了我在面对困难时如何沉着应对，在面对一个陌生的课题时，如何寻找突破点。

感谢同在华为实习的同学应学义，江涛，顾客，董湘均，胡小元，李强，杜渐，他们不仅仅是我工作上的同事，更是我生活中的好伙伴。

感谢我的家人，为我完成学业，他们付出了很多。也是他们的鼓励和支持，给了我进取的信心和学习的动力。

感谢所有帮助过我、关心我的师长、亲友、同学们。

# 附录一 攻读硕士学位期间完成的论文列表

[1] 王子敬 王勇 徐侃烈，基于 Mediator 模式的参数控制法. 江南大学学报自然科学版
[2] 褚雄 王子敬 王勇，一种基于 FPGA 的 DES 加密算法实现 江南大学学报自然科学版

# 附录二 攻读硕士学位期间参加的科研项目列表

[1] 华为公司合作项目，3G 移动网管研究与开发
[2] MPEG4 图像编码器实现