计算机科学 1999Vol. 26№. 5

STARS 开放体系结构的可复用资产库框架

An Introduction to the Asset Library Open Architecture Framework(ALOAF)

常继传 梅 宏

(北京大学计算机科学技术系 北京 100871)

Abstract Reuse library mechanism and automated reuse tools can greatly facilitate the acquisition, integration, classification, identification, and management of software components. In the fall of 1992, STARS project proposed the ALOAF report to address the problem of resource sharing and seamless interoperability between reuse libraries. This paper introduces ALOAF report version 1.2, including ALOAF Asset Library Framework Reference Model(ALF-RM)and ALOAF Specifications.

Keywords STARS, ALOAF, Asset Library Framework, Interoperability, Meta-model, Data model, Service model

一、前言

近年来由于面向对象等新技术的支持，软件构件技术已经成为热点。人们认识到为了充分利用构件和完成大量构件的生产、分类、检索、集成和维护任务，构件库及其相关工具将变得十分重要。国内的相关研究工作已经展开，青鸟工程在"九五"期间的主要任务之一便是研究基于"构件—构架"模式的软件复用技术，为我国软件企业提供良好的技术装备。国外对此也有巨大的投入，在美国军方与政府部门发起的项目中，到 93 年底已经有了 CARDS/ASSET/DSRS(合称 CAD)等大型可复用资产库系统，并且开始考虑资产库间的相互合作。

由 DARPR 发起，美国军方、CMU/SEI 和 MITRE 提供支持的 STARS 项目早在此前便开始考虑在开放体系结构的可复用资产库之间共享资源和无缝互操作的问题，并于 92 年提交了 ALOAF (开放体系结构的可复用资产库框架)Version 1.2 版本。这一报告体现了 STARS 对可复用资产库系统的认识，给出了一个资产库框架的参考模型(不妨类比 ISO 的 OSI-RM 和 ECMA 关于软件工程环境的 Toaster 模型)，并对此给出了 ALOAF 规约来作为该参考模型的实例，由此证明以公共元模型为基础在资产库之间交换信息和创建易于移植的复用工具是必要的和可能的。

在将与复用相关的方法和技术并入软件工程主流方面，STARS 项目作出了相当的努力。在 87 到 92 年间，STARS Foundations，STARS Prime 及其它复用项目都有成功的结果。1990 年秋 ALOAF 有了初始的构想，几个月后成文。1991 年 1 月，三个 STARS Prime 的参与者—波音(Boeing)公司、IBM 和 Paramax(前身为 Unisys 防务系统公司)全部参加到了 ALOAF 的工作中，ALOAF 是这三家公司及其它参与者合作的结果。

ALOAF 的初始想法考虑了 STARS 复用库的短期目标和长期目标，短期内尽快在若干不同的复用库之间共享可复用资产，长期目标则是达到异质资产库之间无缝的互操作，包括共享资产及其描述以及使一组丰富的可复用资产库工具易于移植。ALOAF 文档将跟踪软件复用方面的新进展并不断进行改进。

本文便是对 STARS 项目的 ALOAF 作一总结介绍，以供国内同行参考之用。

二、ALOAF 的背景

STARS 对资产(Asset)的定义是：任何在当前

[1] 不同的项目对资产或软件资产有不同的定义，比如青鸟构件模型中的构件(Component)是指软件产品中可以明确辨识的组成成分，可分为分析件、设计件、代码件、测试件等。这并不妨碍我们在各自的语境中讨论它们。

或将来对于开发和维护软件系统的企业和组织有价值的信息单元"。这一概念比狭义的软件构件意义更为广泛。资产可以包括软件工作产品、软件子系统、源代码构件、专家联系名单、构架、领域分析、设计、文档、个案分析、经验教训、研究成果以及有创见的软件工程思想与表述等等。可见 STARS 的 Asset 定义不仅仅限于开发与维护系统用到的某个软件生存周期的成果或过程，这一点与 Mili 的定义稍有不同。STARS 认为围绕资产有以下几样是可复用的：资产、资产的描述信息、复用工具和支持子系统以及一组一致的约定和接口。

框架(Framework)是用于支持或容纳的骨架结构。在复用文档中它是界定被讨论概念和促进技术转变与演进的概念结构。一个资产库框架(ALF)定义了一张蓝图，据此可以实现资产库系统和创建基于复用的库工具；而对于开放体系结构的资产库框架 ALOAF，这张蓝图是共有的而非专有的。

ALOAF 主要包括三大部分：

· 由资产库框架中的共同元素和共同服务构成的参考模型；

· 在参考模型上完全支持资产互换和资产描述的 ALOAF 服务规约；

· 为在不同资产库间互换资产而引入的数据模型和格式约定的规约。

2.1 ALOAF 与 STARS 的复用观点

为了更好地理解 ALOAF，有必要先了解 STARS 是如何认识复用的。

STARS 项目的观点是"软件系统的开发将演化成一种过程驱动的、特定于领域的和基于复用的技术支持范型"。其中"特定于领域的和基于复用的"开发范型显式适用于 ALOAF，应用程序以基于资产的方式而非从头新建的方式被构造出来。可复用资产被组装到应用程序中，它应该基于特定领域的体系结构并遵照开放的接口标准。

为了满足发起者美国国防部的要求，STARS 的一个关键技术目标是由在市场上可以购买到的软件工程环境框架和 CASE 工具构造软件工程环境(SEE)。STARS 认为资产库系统包括可复用框架元素和复用工具。它希望构造出遵照某一开放体系结构的模块化的复用工具与复用框架，并由此构造出开放的资产库框架。环境装配者从购买的或自有的工具和系统中选取符合 ALOAF 标准的工具，这些工具可以互操作并且与环境框架相容，因此集成得到的软件工程环境是基于标准的和标准驱动的，易于剪裁，易于修改，更为可靠。

提出 ALOAF 并不是要扼杀新的复用方法和复用环境，而是希望不同的复用项目从各自的工作中受益，包括：

· 共享和互换资产及其描述信息，由此可以利用最新最好的资产来创建应用程序；

· 符合 ALOAF 服务接口的库工具易于移植，资产库系统也就易于用外来的复用工具扩充。

基于此，复用技术和方法以及复用元素(构件)可以作为整体迅速扩充，从而加快传统的软件开发向基于复用的软件开发的转变。

除了 ALOAF 之外，STARS 还提出了"STARS 复用概念"文档，并参加了 RIG 组织，前者声明了 STARS 对跨系统跨生存周期的资产复用的观点和对复用目标、复用过程的认识，ALOAF 与该文档相一致并促使其实现。RIG 组织的目标是方便由政府发起的软件复用库之间的互操作，STARS 参加 RIG 后，ALOAF 也成为 RIG 的候选标准。ALOAF 积极参加了标准制定工作，ALOAF 的元模型成为 RIG 提出的 UDM 元模型的基础[5]，ALOAF 的 CDM 也被 RIG 的 UDM 参考。今后 STARS 将遵从 RIG 的标准并使之表现在 ALOAF 中。另外 STARS 还完成了库间互操作演示(Library Interoperability Demonstration)[6]，这一体系结构原型发展了 ALOAF 对资产库的认识，表明未来的复用库应该分解为显式的三层，即：

· 存储层：安全地保存资产并提供存取资产的功能；

· 编目层：对存储在不同库中的资产编目和作增值索引；

· 用户界面层：提供给用户访问不同资产目录的用户界面。

库间互操作演示标志着资产库由单独的封闭式构架建成的时代的结束，预示围绕资产和资产库有了新的商业机会。

2.2 ALOAF 与软件工程环境(SEE)的关系

资产库是软件工程环境的若干子系统之一，作为子系统的资产库提供的功能应该与 SEE 框架的参考模型相一致。开放体系结构的资产库框架(ALOAF)与 ECMA 的 SEE 框架有两个重要的不同：

1. ALOAF 关注资产库框架而非更广泛的 SEE 框架；

2. ALOAF 不仅包括资产库框架的参考模型、

而且有服务规约和数据描述规约。

ALOAF 在资产库和复用服务方面补充了 SEE 参考模型。在某些情况下，ALOAF 与对象管理服务（OMS）有密切关系。从 SEE 的角度来看，ALOAF 的服务是 SEE 之上的一层，资产库系统利用了 SEE 提供的服务和工具（比如用户界面、OMS 和数据传输子系统）。当然，ALOAF 并不一定要建立在 STARS 的软件工程环境之上。

2.3 ALOAF 基于的几个前提和假定

ALOAF 的需求建立和开发基于若干重要前提和假定：

· 国防部的需求—国防部要求采用基于复用的软件开发和维护，使软件质量和生产率达到 90 年代水平。具体涉及到在地理上分布的组织，他们使用完全不同的软件工程环境，各个组织之间共享可复用资产。

· 资产库技术尚不成熟，已有的资产库机制都是各自独立开发、互不关联的，几乎没有标准化的努力。

· 资产不只包括一般意义的源代码构件，也包括以下信息：

◇其它形式的可复用软件产品，如需求规约、构架、设计和测试件等

◇应用领域知识，如模块、数据字典、算法等

◇过程定义，如管理资产库或开发应用系统的过程模型

◇决策依据，如系统中为何要包含某一特性、服务、目标或算法；为什么选择某一体系结构或设计而不是另外一个等。

· 资产库中的数据依照数据模型来描述，并非所有的资产库都使用同一数据模型，同一资产库的数据模型也会随时间而变化。

· 由资产库系统管理资产，访问资产需要经过资产库系统。

· 资产库用户的需求—用户在库中查找各种资产来支持软件系统某个生存周期阶段的活动。用户可能要访问多个资产库以使复用的数量和有效性尽可能大。理想情况是用户通过一个最喜爱的界面访问所有资产库，不需要关心它们的位置和特性。

三、STARS 的资产库框架参考模型（ALF-RM）

3.1 资产库的概念

STARS 提出的参考模型表达了对于资产库及其组成元素的一般性观点，并为此建立了基本术语。下面图 1 是参考模型的最基本示意图，从中可以看到若干重要的概念及其相互关系。

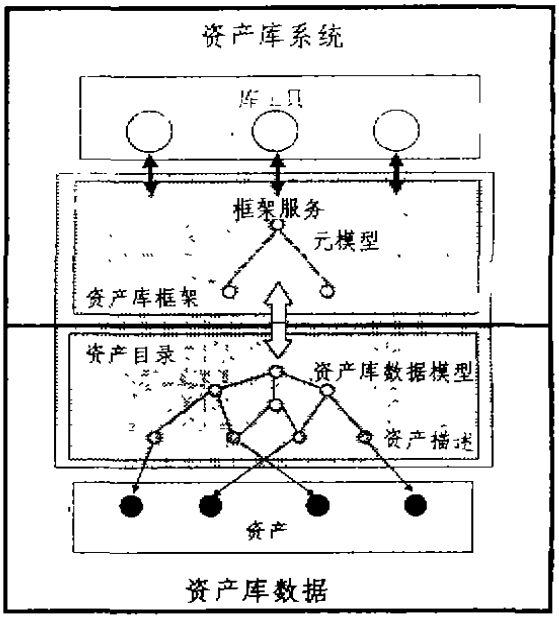


图 1 资产库

图中最外层方框表示资产库，它是一种组织、收集、访问与管理若干资产的手段。资产库由资产库数据和资产库系统组成，图中以横线分隔二者。资产库数据包括资产本身及其描述性信息和组织性信息，这些信息总称为资产目录（图中阴影的下半部分）。其中描述性信息称为资产描述，组织性信息称为库信息模型。

资产库系统由资产库框架和资产库工具组成，它提供了一组面向库的定义和操纵资产库中数据的机制，资产库框架 ALF 提供了基本的面向库的操作（图中"框架服务"），这些操作是按照由元模型定义的某种数据组织技术来定义、创建、操纵和管理资产目录所必须的。库工具提供了若干有结构的操作，高层工具和用户通过框架服务来访问资产目录并间接访问资产。ALOAF 主要关心资产库框架和资产目录，并不关心资产本身和建立在框架服务上的库工具。

在图 1 中，元模型用于定义和界定库中数据模型被创建的方式。图中的白色双箭头表示出元模型与数据模型间的直接关系。一旦给定了数据模型，资

产描述将反映由它所定义的资产目录的结构，资产目录决定了如何描述资产以及资产描述信息的格式。

3.2 数据建模的概念

资产库框架包括三个数据建模层次，元模型层、数据模型层和数据层。

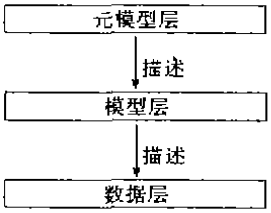


图 2 数据建模层次

图 2 中显示了三个层次之间的关系，其中以元模型层最一般，数据层最特殊。以互交换和互操作为目标，ALOAF 尤其强调元模型与数据模型的概念，数据建模是 ALOAF 参考模型中非常重要的一个部分。

3.2.1 数据建模的层次 元模型层由各种数据建模技术组成。在组织复杂度不同的数据时，使用何种特定的数据建模技术是一个基本的选择。ERA 建模、数据流建模和面向对象建模都是常见的通用建模技术。它们各有变种，可以以非形式化描述、形式化语言、图形方式（以特定图标代表特定含义）等不同方式表述。一个元模型可以是任何一种特定的数据建模技术，它提供了表达一类数据模型的手段。

模型层由数据模型组成。一个数据模型符合某一元模型，并描述了某一特定应用领域或特定组织的数据。数据模型的同义词是"模式"和"信息模型"。图 3 是一个符合 ERA 元模型的数据模型的例子，表达该模型的语言（即元模型）由方框、箭头和图例等组合而成。数据模型本身由"资产"组成，"资产"有"唯一标识"、"资产名"和"版本号"3 个属性。资产由若干"文件"组成，每个"文件"有"唯一标识"和"文件名"2 个属性。该数据模型可以进一步精化，比如可以确定属性的数据类型和限定取值范围。

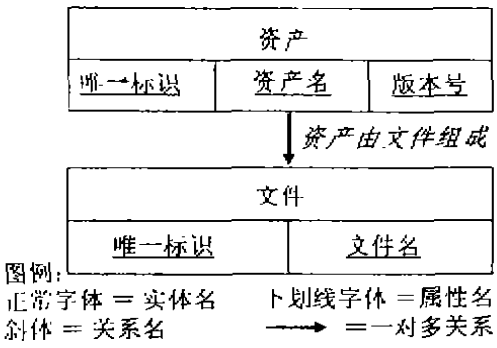
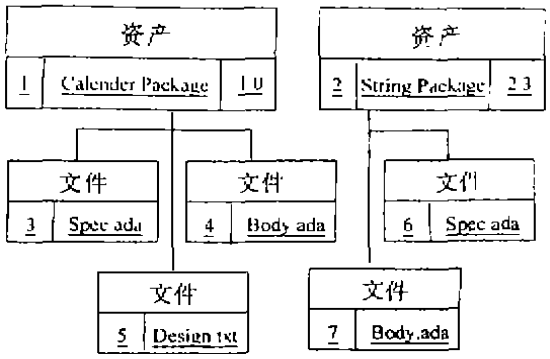


图 3 ERA 数据模型

数据层由实际存储的信息组成。信息被组织成符合某一数据模型。图 4 中显示的是符合图 3 所示数据模型的一些数据实例。



图例：——= 关系实例
正常字体 = 实体名 下划线字体 = 属性值

图 4 ERA 数据实例

3.2.2 数据建模与资产库之间的关系 资产库维护了一组资产并提供一种或多种分类模式，复用者可以借此在库中定位某个资产。资产库也向复用者提供每个资产的信息，例如摘要和发布日期。库中维护的资产信息（包括支持分类的信息和直接向复用者提供的信息）是按照资产库的数据模型来组织的，数据模型又依照了元模型，它们在资产库中的角色见图 1。

因为面向的用户以及关注的焦点不同，各个资产库通常有不同的数据模型。最普遍的情况是数据模型被"硬写"进创建资产库的程序中，当改变所维护的数据的性质时要重新编译或进行数据库的格式转换。通常通过资产库支持工具的数据定义功能间接理解或非形式化刻画这些数据模型所符合的元模型，某些库系统，象 Paramax 的 RLF 和 SAIC 的 AMS，提供了更为通用的数据建模功能，允许不同的安装版本选择不同的数据模型，这就能够更好地支持特定领域的资产库。

目前大多数资产库系统作为独立的资源运行，用于分类、导入和保存新资产的工具，用于浏览和检索资产的工具以及管理资产库的工具都是库系统的组成部分。这些工具可以认为数据模型是确定的，因为资产库（而并非数据库）中仅有一个数据模型。即

使是支持任选数据模型的库系统也得益于数据建模所带来的"共性"—所有库共享了同一元模型,该元模型是资产库开发者所私有的。而异质资产库间互操作提出了新的要求—与不相关的库互换资产,于是资产接收方的库系统必须理解收到的信息,换言之,现在需要理解其它资产库的数据模型。

在理论上可以构造一个标准的固定数据模型的资产库,要求所有资产库都符合它,但是这一方法局限性太大。另外的方法是接收方以某种预定义的方式"获知"发送方的数据模型,但是这需要与所有的发送方经常交互以便跟踪和维护各自数据模型的变化(ALOAF 假定资产库数据模型会变化)。因此只有让发送方的信息中包括所发数据的数据模型或者让接收方动态地向发送方请求获知有关信息,这都要求接收方能读懂并解释有关数据模型信息,可见,互操作的系统之间必须达成某种共识;换句话说,共享元模型是资产库互操作的前提。

在更高层次上,共享的元模型可用于描述不同资产库的数据模型,因此可以用一个标准的元模型来支持服务的定义和数据交换。PCTE、ATIS(一个工具集成标准)以及 ANSI 的 IRDS(信息复用字典系统)作为不同领域的标准都定义了操作符合某一标准元模型的数据的服务。CDIF 和 IRDS 都支持基于元模型的数据格式,同样的机制也可用于资产库系统。标准的元模型是完全不同的资产库间统一理解数据模型(再由此理解数据)的手段。

3.3 服务模型

STARS 的 ALF 服务模型定义了库框架向复用工具提供的服务及其分类,同类服务与各类服务间的关系,并给出了复用工具开发者与库框架提供者之间的标准接口和术语记号。ALF 服务模型应该放到整个 SEE 框架的服务模型中考察,因此有必要介绍 ECMA/NIST 的 SEE 参考模型。ECMA 的模型使 SEE 服务间以及 SEE 服务与 SEE 工具间的接口

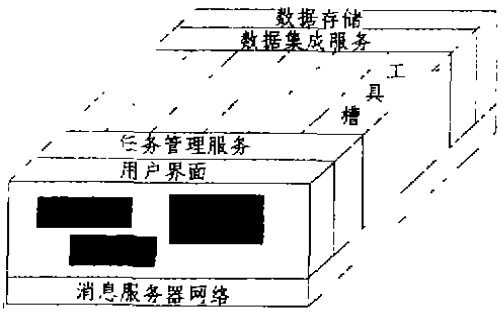


图 5 ECMA 的面包机模型

规约成为可能,使在完整的 SEE 所提供的浩繁的服务中找到某个服务变得简单,并能帮助我们评价和比较不同的 SEE 对互操作性与易移植性的支持。图 5 为著名的"面包机(Toaster)"模型,从中可以看到软件工程环境所提供的服务的分类与其间的关系。

ECMA 的 SEE 模型将服务分为以下几类:

- 数据库服务
- 数据集成服务
- 工具
- 任务管理服务
- 消息服务
- 用户界面

资产库框架在宿主软件工程环境或宿主操作系统中为一类应用(尤指资产库及相关工具)提供服务,实现库框架服务需要用到宿主环境提供的底层服务。简单的软件工程环境只提供少量基本服务,实现 ALF 较困难;而在基于 OMS 的功能丰富的软件工程环境中,ALF 的实现就简单得多。SEE 的某些服务,如用户界面和任务管理服务可以直接被资产库工具所用,而消息服务可用于提供分布的库系统之间无缝互操作所需的通信协议。但可能还需要开发高层的资产库协议来刻画库系统的语义信息。

3.3.1 ALF 服务的语境 虽然有许多方法为 ALF 服务分类的定义和作用域界定语境,但是将资产库与数据库相类比可以得到一些有趣的相似点,对 ALF 也会有更深入的理解。

资产库包含了大量的信息,它主要是一个软件信息数据库,这些信息包括:资产、资产描述信息、资产库数据模型和描述数据模型的元模型。

ALF 的数据建模服务恰好与关系型数据库中创建和修改表的服务相似,它们处理的都是库中数据的基本结构而非数据本身。这类操作也同样带有与修改表结构操作类似的危险性和潜在的全局影响,因此往往只有管理员有权使用。资产库可以在 DBMS 之上实现其目录,在一个形式化的配置管理系统(比如 SCCS 或 CMS)中保存资产。未来的资产库设计可能基于一个大型的对象管理系统、与开发工具紧密集成,这样资产也许将不会再以单个可标识的文件的形式存在,"资产"和"资产描述"之间的分界将变得模糊。

3.3.2 ALF 服务的分类 图 6 显示了 ALF 服务、ALF 上建立的工具以及复用者之间的关系,该图可以看作是图 5 中的三维模型的二维截面图。

ALF 服务分为以下几类:

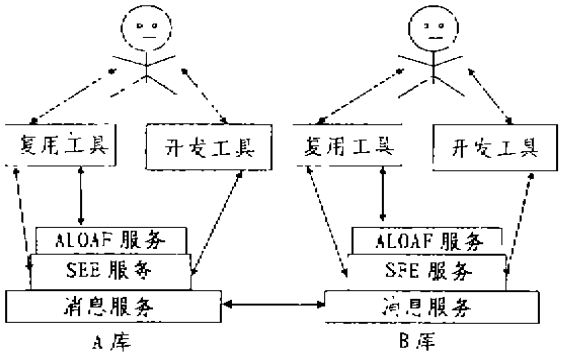


图 6 SEE 与 ALF 服务间的关系

◇会话服务：管理着与资产库的连接，支持用户及代表用户操作资产的复用工具在不同场所之间传输资产及其描述，还包括所有与设定用户身份与权限有关的服务，并支持记录的保留、用户与系统间的消息通告等。

◇库管理服务：在库的操作环境中管理和操纵资产库，包括创建新库、删除库、打开和修改库以及关闭资产库等操作。

◇数据模型服务：按照特定的元模型管理和操纵资产库的数据模型，包括读、写、更新、删除以及导入/导出数据模型等。

◇资产描述服务：管理和操纵个体资产描述，包括对资产描述消息的增、删、改、查和导入/导出等。

◇查询服务：提供灵活的查询机制在资产库中定位资产，资产库提供了在资产描述信息的属性之上的代数操作，在已知数据模型的前提下查询资产库以找到与查询条件最佳匹配的资产集合。

◇资产处理服务：一旦用户找到感兴趣的资产，提供为获取资产并充分利用资产而需要的信息和处理资产的机制。

◇度量服务：提供关于资产库的结构和使用情况的度量信息，也包括控制统计信息的收集和存放的相关服务。这些度量信息可对用户和库管理员提供有效的帮助。

◇访问控制服务：管理用于确定用户对库数据和服务的访问权限的相关信息，帮助维护对不同资产和不同用户在法律或政策方面的各类限制。该类服务与传统的 SEE 访问控制表(ACL)不同，能在小到数据模型的属性、大到复用者角色的不同粒度上进行访问控制。

3.3.3 所需的 SEE 服务支持 以下罗列 ALF 所需的 SEE 服务种类

◇数据存储与永久化服务。

◇关系服务：定义和维护类似于 ER 模型中的类型化的、属性化的"关系"。

◇名字服务：在系统生成的 UID(唯一标识)与用户可读的助记标识符间进行动态转换。

◇分布与位置服务：支持软件工程环境及其对象的分布。

◇数据的事务处理：保证用户工作的完全原子性。

◇并发服务：支持对象的同时访问。

◇OS 进程服务：对活动实体的基本支持机制、管理实体的执行状态并在实体执行时进行监控。

◇归档服务：实现在线存储与后备存储间的映射。

◇备份服务：发生媒质故障时，提供恢复机制并利用日志重新完成已提交的事务。

◇数据变换与互交换服务：比如模型描述的变换，内、外存存储格式间的变换。

◇状态监控与触发器服务：即"历史服务"，对实体的状态变动提供信息记帐的功能。

◇版本服务：对实体提供版本管理功能。

◇安全服务：SEE 安全服务的子集，用于限制对对象及其服务的访问。

◇工具登录服务：将工具及其可访问的对象类型进行关联，以实现一定的安全性。

四、ALOAF 规约

ALOAF 规约是前述资产库框架参考模型的 STARS 版实例。该规约包括元模型规约、服务规约、构件互交换规约以及 ALOAF 编程接口的 Ada 规约。

4.1 元模型规约

ALOAF 提出的元模型基于 STARS 资产库机制的开发经验和有关数据建模的文献。该规约分为核心元模型和元模型扩展两部分。前者包括了所有 ALOAF 实例都应支持的基本数据建模技术；后者也十分重要，但限于技术上与经济上的困难并不要求全部支持。

标准的资产库元模型有三大好处：首先它是独立于资产库对库中数据模型进行表示的基础，二是它支持独立于服务规约的数据模型的开发，二是它

为开发资产库数据模型的人们提供了基本术语和概念框架。

资产库数据模型的基本组成结构是"类"，有"特化"和"例化"两个基本概念。特化是类与类之间的基本关系，例化是由资产库数据模型的元素（类）构建资产描述（对象）的技术。类由属性组成，对象由域组成，对象是类的实例，对象的域与类的属性相对应，类定义了对象的结构，属性限制了相应域的取值范围。

要注意的是这里的类、对象、属性、特化、例化等概念与OO观点相似，不的同是OO所见的类是有相同结构和行为的对象的集合，而ALOAF的元模型仅为元数据模型，不含对象的行为成分。属性定义了这种相同的公共结构；每个属性代表了一些该结构中可访问的个体，一个类的属性数不受限制，确定一个类所需的信息是类名和父类。

属性有单值与多值之分。

一个单值属性由属性名与域类型决定；单值属性的域类型有：Boolean（布尔型）、Integer（整型）、Float（浮点型）、Time（时间型）、String（字符串）和Enumeration（枚举型）。枚举型属性还要定义Vocabulary（词汇表），即标识符序列，该序列的次序由定义者建立，在一个词汇表中标识符不可重复。

一个多值属性由属性名、上界、下界和域类型决定；上界、下界分别表示相应于该属性的域中取值个数的最大值与最小值（下界非负）。多值属性的域类型有：Keyword（关键字）、Facet（刻面）和Link（关联）。关键字从任何可读的标识符中取值，刻面取值于自定义的词汇表[2]，关联取值于对象的UID。刻面型的属性需要知道词汇表的内容，关联型的属性需要知道目标类。多值属性的域中有一组值，这些值类型相同，取值不同。

数据模型中的类被组织成特化层次结构，每个类都参与特化关系（也叫父子关系），形成的树形结构中仅有唯一的根类。特化关系的语义为子类继承所有祖先的属性。核心元模型的特化关系仅限于单继承，扩展元模型则允许多继承。

每个对象由一个类例化而来，且仅为一个类的实例（单实例）。例化所需的信息为对象名和初始域值（对象每个域的初值），在扩展元模型中，一个对象可以是多个类的实例（多实例）。

4.2 服务规约

ALOAF的现行版本给出了前面描述的几乎所有服务分类中核心服务的详细定义，并特别着重于数据建模服务和资产描述服务。服务的描述包括如下方面：·接口；·基本行为；·出错条件和处理。

接口除了以语言无关的形式刻画外，也有Ada编程接口（限于篇幅不再表述，有兴趣的读者可参见文献），ALOAF给出了一个最小的核心服务集合的规约，对于某些欠缺之处（比如数据建模与资产描述服务必须加入导入/导出能力，在资产处理服务类中加入具体服务等等）将逐步进行改进和扩展。

4.2.1 服务的基本特性

ALOAF中的"对象"专指类的实例，因此用"实体"来指代ALOAF服务所操纵的元素。ALOAF服务规约中最常用的类型叫做"句柄"，它是关联到所希望访问或修改的实例的一种方式，它与编程时用到的"指针"类似，但句柄不仅包括指向实体的指针，也可以包括该实体的其他信息，所以二者并不完全相同，许多服务与句柄管理有关，只有在句柄"打开"后才能修改一个实体。另一个公用的类型是名字类型，实体被赋以可读的名字来代表真实世界中的概念或关系。如果服务处理若干实体，则用到名字集合类型，一个集合中的名字可以逐个处理，各自转换成打开的句柄并对应到相应的实体。

服务的成功或失败等信息作为状态参数被包含在服务接口中，其类型为状态类型。ALOAF服务规约中指出的错误被认为是违反了服务语义的高层指示信息，不考虑形参/实参类型不匹配等简单错误。除了返回布尔类型的服务之外，服务有OUT[1]参数指示其执行状态。当服务失败则其他OUT参数无意义。

我们容易从约定的服务名和参数名的字面含义上得知其语义，比如：以Open_、Close_、Get_*_Name等作为前缀的名字有相似的语义。Open将一个合法的名字转换为打开的实体句柄，Close则返回为其分配的资源，Delete是将其从永久存储中移出。

4.2.2 服务间的关系

ALOAF实现所管理的信息被组织为图7所示的信息层次结构，这一结构定义了静态的（物理的）资产库系统交互语境，也显示了定义库系统交互语境的正交的ALOAF活动层次。

---

[2] ALOAF定义的刻面仅有取值范围的限制，其它复用项目的刻面定义可能更为严格。

[1] 类似Ada的IN和OUT参数。

信息层次结构的最高层是数据库，其中包含任意数目的数据模型和库(Library)。数据库不特指某种物理表示机制，它可以是文件系统、OMS 或关系型数据库，也可以是分布式的，数据库定义了名字空间的边界，所有的库和数据模型在数据库中有唯一标识。资产库(Library)必须包含一个相关联的数据模型，但是数据模型却可以存在于所有库之外。类存在于数据模型中，对象在资产库中，资产被描述为指向到库中的对象。在信息层次中没有元模型，因为元模型是描述其它 ALOAF 实体的手段，本身并不参与操作，也不被物理地表示在库系统中，一个数据模型有唯一的存在，它不可以同时与两个库关联，也不可又在库中、又在库外(复制的数据模型有不同的 UID)。只能修改未与具体库关联的数据模型。

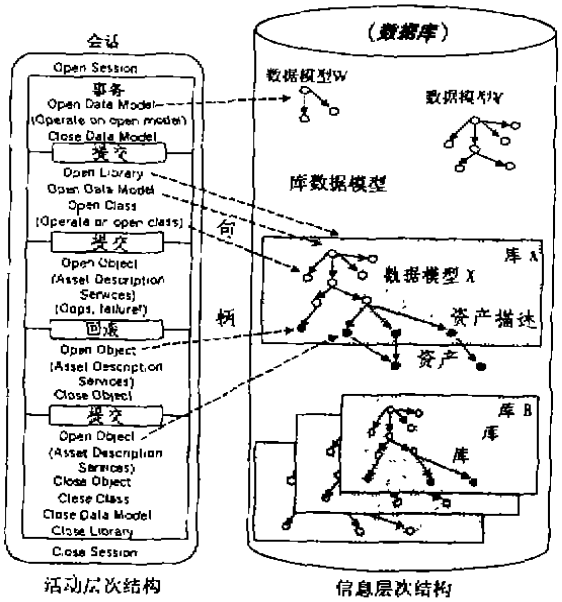


图 7　活动层次结构和信息层次结构

信息层次结构中的另一实体为用户数据库，它在概念上与库和数据模型处于同一层次，其中保存用户及其权限信息。用户数据库独立于库和数据模型，不必符合 ALOAF 元模型，其内部结构与具体实现有关。

每个实体为其中所包含的实体定义了语境。一个实体的句柄是在给出实体的语境后由 Open 操作获得的，得到的句柄可用于为下一层实体建立语境，再一次获取下一层次的实体的句柄。

活动层次结构的最高层是"对话"，对话映射到单一的数据库，该数据库的名字是 ALOAF 实例已知的。在一个对话中选择数据库时，静态的和动态的两个层次结构便相交了。实现上 ALOAF 服务用对话句柄而不是数据库句柄标识最高层语境。

每个对话包括一系列的事务，事务有提交和回滚操作。事务中又各自封装了若干服务序列。用户与 ALOAF 实例以"打开对话"来建立动态语境。每个对话开始时自动建立一个事务，一个事务的终结意味下一事务的开始，直至整个对话结束。由此可见 ALOAF 的事务语境与对话语境是相同的，不必再有事务句柄。这一简化导致很大的局限性：虽然对于同时打开的库和数据模型的数目没有限制，但活动必须限制在一个事务语境中发生。克服该局限是未来进一步扩展 ALOAF 服务规约的主要原因。

4.2.3　**脚本实例**　限于篇幅，这里不再将 ALOAF 服务规约逐条列出，而只是列出其分类，它们是：

◇Session Services
◇Library Management Services
◇Data Model Services
- Model Definition Services
- Object Class Services
- Attribute Services
- Instantiation Services

◇Asset Description Services
- Object Services
- Field Services

◇Query Services
◇Asset Processing Services
◇Metrics Services
◇Access Control Services
◇Basic Services
- Iterator Services

以下举一个脚本实例说明如何用规约中的服务来扩展数据模型，从中可以看出 ALOAF 服务规约的风格和用法。首先定义一些要用到的数据项，本例中以_id 为后缀的参数是各种句柄参数，以_status 为后缀的是状态参数(不再声明)，双引号括起来的是用户名、口令字、数据模型名、类名、库名等字符串：

```
session_id:Session_Handle_Type
cdm_id,ldm_id:Model_Handle_Type
example_lib:Library_Handle_Type
process_model_class, requirement_class, objet_
        class,asset_class:Class_Handel_Type
```

脚本：

```
1. Open_Session("User_Anonymous","Password_STARS",session_id,session_status)
2 Open_Data_Model(session_id,"Common_Data_Model",TRUE[4],cdm_id,model_status)
3. Copy_Data_Model(cdm_id,"New_Model_Name",ldm_id,model_status)
4 Get_Root_Class(ldm_id,object_class,class_status)
5. Create_Class(ldm_id,object_class,"ProcessModel",process_model_class,class_status)
6. Open_Class(ldm_id,"Asset",asset_class,class_status)
7 Create_Class(ldm_id,asset_class,"Requirements",requirement_class,class_status)
8 Create_Library(session_id,"New_lib","New_Model_Name",example_lib,lib_status)
9. Close_Data_Model(cdm_id,model_status)
10. Close_Data_Model(ldm_id,model_status)
11 Close_Library(example_lib,lib_status)
12 Close_Session(session_id,TRUE[5],status,session_status)
```

结果是在CDM(见"资产互交换规约")中扩充了两个新类，process_model_class 是 object_class 的子类，requirement_class 是 asset_class 的子类。扩充后的新数据模型与一个新创建的库 example_lib 相关联。

4.3 资产互交换规约

ALOAF 在参考模型中提出了资产互交换和系统互操作的需求和以共同的元模型为基础的原则，在服务规约中将资产互交换所需的数据建模服务作为重点，而且提出了公共数据模型 CDM 和与之配合的资产互交换描述语言作为资产互交换规约的一部分。该模型和语言足以达到 ALOAF 的简单互交换能力的目标。STARS 将 ALOAF 交给了 RIG 组织，今后会参与但不再独自进行互交换和互操作方面的努力。了解 RIG 的读者会注意到 CDM 与 RIG 的 UDM 和 BIDM 极其相似，对于 CDM 的具体描述可以参见有关报告或直接参考 UDM。ALOAF 关于互交换的长远目标是开发灵活的机制以有效地满足异构分布式资产库的互交换需求，这将是基于标准元模型(ALOAF 的元模型规约)和独立于资产库的数据模型表示和数据表示方法(即资产互交换数据格式 AIDF)。在现有的 ALOAF 文档中未包括它们，附录中列出了有关的标准。

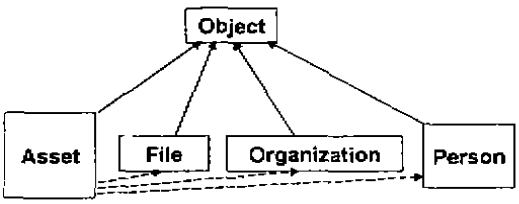


图 8 CDM 类层次

4.4 与 ALOAF 规约的符合度

STARS 认为与 ALOAF 的符合程度可以从资产库系统所提供的服务质量、服务数量和服务模式(Mode)三个维度考虑，每个维度上定义不同的符合度。

服务数量为最低符合度的系统不提供 ALOAF 接口，但支持导入/导出，其它符合度的资产库则应完全支持核心服务(包括数据建模服务和资产描述服务)以及不同程度的扩充。服务质量则考虑了效率、可靠性、可用性和对环境的依赖程度等非功能因素。服务模式分为过程模式和协议模式，前者是当前 ALOAF 规约所采取的形式，通过语言联编的方式提供服务，要求客户应用程序在运行时被链接和装入；后者是 ALOAF 今后的目标，它将以消息连接的通信协议来处理应用程序与资产库系统间的客户/服务器关系。

ALOAF 打算再将不同维度的符合度综合起来，分成若干总体符合度，已提出的两个级别是：

◇资产互交换级：(数量上)支持资产导入/导出，(质量上)要求服务的正确性。

◇核心服务支持级：(数量上)至少支持 Ada 编程接口，(质量上)要求服务的正确性、可用性及较好的效率，(模式上)以协议模式提供服务。

对于与 ALOAF 的符合程度的评价、验证和保证是 STARS 今后的工作之一。

[4] 该布尔参数表明用户是否具有写权限。

[5] 该布尔参数表明是否在关闭本次会话时提交事务。

结论：STARS 项目考虑了在资产库之间共享资源和实现无缝互操作的问题，提出了开放体系结构的资产库框架(ALOAF)，ALOAF 强调数据建模的作用，考虑元模型层、模型层和数据层三个层次，认为统一的数据元模型是互操作和数据共享的基础。ALF 在宿主 SEE 环境上为资产库机制和相关的复用工具提供服务接口，以这组标准的服务为基础实现资产库间的互操作。

附录 1 头字母缩写

□ASSET Asset Source for Software Engineering Technology

软件工程技术的资产源

□ALOAF Asset Library Open Architecture Framework

开放体系结构的资产库框架

□ALF Asset Library Framework

资产库框架

□CARDS Central Archive for Reusable Defense Software

可复用防务软件的中央档案库

□CASE Computer-Aided Software Engineering

计算机辅助软件工程

□CDIF CASE Data Interchange Format

CASE 数据互交换格式

□CDM Common Data Model

通用数据模型

□DSRS Defense Software Repository System

防务软件库系统

□ECMA European Computer Manufacturing Association

欧洲计算机制造商联合会

□ERA Entity Relationship Attribute

实体关系属性

□OMS Ojbect Management System

对象管理系统

□PCTE Portable Common Tool Environment

通用可移植工具环境

□RIG Reuse(Library)Interoperability Group

复用(库)互操作小组

□RLF Reusability Library Framework

(Unisys 的)复用库框架

□SEE Software Engineering Environment

软件工程环境

□STARS Software Technology for Adaptable, Reliable Systems

用于易修改的可靠系统的软件技术(项目)

附录 2 参考文献

1 STARS. Asset Library Open Architecture Framework Version 1.2: [Informal Technical Report, STARS-TC-04041/001/02]. 1992

2 Mili Hafedh, et al. Reuse Software: Issues and Research Directions. IEEE Transactions on Software Engineering, 1995, 21(6)

3 Sindre Guttorm, et al. The REBOOT Approach to Software Reuse. System Software, 1995, 30: 201～212

4 青鸟工程项目组. 青鸟构件模型:[青鸟工程项目组技术报告]. 北京大学计算机科学技术系, 1997

5 RIG. Basic Interoperability Data Model: [Technical Report, RPS-0001]. Reuse Library Interoperability Group, 1993

6 Berggren P. Library Interoperability Demonstration. In: Proc. of the DARPA Software Technology Conference, April 1993

附录 3 ALOAF 所涉及的标准

1 ANSI 的 IRDS(Information Resource Dictionary System)标准

2 EIA 的 CDIF 标准

3 ECMA 的 PCTE 标准

4 与构件互交换相关的标准有 ANSI IRDS、EIA CDIF、IEEE P1175 和 RIG 的 BIDM 与 CDM (BIDM 于 1995 年成为 IEEE 标准 1420.1)