ICS 35.080
L 77

中华人民共和国国家标准

GB/T 25644—2010

信息技术
软件工程　可复用资产规范

Information technology—
Software engineering—Reusable asset specification

2010-12-01 发布　　2011-04-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　次

前　言

本标准的附录A为资料性附录，附录B为规范性附录。

本标准由全国信息技术标准化技术委员会提出并归口。

本标准主要起草单位：上海计算机软件技术开发中心、上海宝信软件股份有限公司、万达信息股份有限公司、上海市软件行业协会。

本标准主要起草人：宗宇伟、张敬周、葛孝堃、冯惠、郑红、丛力群、张纯、朱三元、李光亚、欧阳树生、钱乐秋、王二卫。

信息技术
软件工程　可复用资产规范

1　范围

本标准规定了可复用软件资产的结构、内容和描述方法。

本标准适用于可复用软件资产的设计、开发、管理、组装、使用等活动，并适用于从事可复用软件资产管理、开发以及实施基于资产开发方式的各类软件组织。

2　规范性引用文件

下列文件中的条款通过本标准的引用而成为本标准的条款。凡是注日期的引用文件，其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本标准，然而，鼓励根据本标准达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本标准。

GB/T 18793—2002　信息技术　可扩展置标语言(XML)1.0

ISO/IEC 19501—2005　信息技术　开放分布处理　统一建模语言(UML)(版本 1.4.2)

3　术语和定义、缩略语

下列术语和定义、缩略语适用于本标准。

3.1　术语和定义

3.1.1

制品　artifact

在软件开发或使用过程中所形成的一种信息的物理件。一个物理上的制品对应于文件系统中的一个文件，一个逻辑上的制品是文件系统中若干相关文件的一个集合。制品的实例如模型、实例或二进制可执行文件等。

3.1.2

问题　problem

在软件开发生存周期中的一个障碍。要满足目标应用的需求，则必须解决(或者避免)开发生存周期中遇到的问题。一个可复用资产完全或部分的解决了软件开发生存周期中遇到的问题。

3.1.3

资产　asset

资产是解决软件开发中某问题的一组制品集合，该问题可与系统制品的演化有关，或直接与所开发系统的领域问题有关。本标准中将资产的制品集合称为解决方案。

3.1.4

可复用资产　reusable asset

可复用资产是对重复出现的问题的解决方案。可复用资产是基于复用思想开发的资产。

3.1.5

白盒资产　white box asset

资产的一种类型，其内部全部可见，并可供查看或修改。

3.1.6

黑盒资产　black box asset

资产的一种类型，该类型资产的内部制品对消费者来说是不可见的。

3.1.7

净盒资产 clear box asset

资产的一种类型，该类型资产的内部制品对消费者是可见的，但消费者不能对其进行任何方式的改变或修正。对外显示资产内部的目的是帮助消费者理解该资产，以便更好地使用和调试。

3.1.8

灰盒资产 gray box asset

资产的一种类型，其内部一部分对用户是隐蔽的，另一部分对用户是可见的、并可被更改的。灰盒资产的可变性介于黑盒资产和白盒资产之间。

3.1.9

构件 component

资产的一种类型。构件是软件系统中具有相对独立功能、可以明确辨识、接口由契约指定、和语境有明显依赖关系、可独立部署的可组装软件实体。

3.1.10

基于资产的开发(ABD) asset based development (ABD)

软件开发过程中的一种方法。基于资产的开发是一套促进资产复用的过程、活动和标准，它没有涵盖软件开发过程的全部。基于资产的开发以体系结构为中心。

3.1.11

提取 harvest

一个 ABD 活动，用于从已有的、未被废弃的系统中创建资产。提取的执行者是资产生产者。提取活动首先是在已有系统中寻找那些有复用价值的组成元素，然后经少量加工将其转换为可复用资产。

3.1.12

应用资产 apply asset

消费者使用可复用资产解决一个问题的 ABD 活动，资产的应用通常要遵循资产规范中的用法指南。

3.1.13

生产者 producer

ABD 中的一个角色，负责可复用资产的创建。生产者可从现有系统中提取资产，或者针对重复出现的问题，从零开始开发可复用资产。

3.1.14

消费者 consumer

ABD 中的一个角色。消费者是一个应用可复用资产的软件开发者。

3.1.15

目标应用 target application

带有可复用资产可解决的问题的一个应用或系统。可复用资产的消费者将该资产应用到目标应用中。

3.1.16

核心 RAS core RAS

可复用资产规范(RAS)的基本描述模型。

3.1.17

剖面 profile

一组语义约束和一个 XML Schema 的集合，用以验证一个实体描述文档。剖面定义了特定类型资产的实体描述文档中哪些信息是必需的，哪些是可选的。

3.1.18

实体描述 manifest

描述可复用资产的结构和组成等信息的一个元信息文档，该文档包含了特定资产的具体描述信息。

按本标准打包的每个资产必须有一个实体描述文档，它是一个经该资产类型的剖面验证有效的 XML 文档。

3.1.19

描述子 descriptor

描述资产信息的一个键/值对。描述子名称是键，通常是人易理解的一两个关键词。值是人易理解的一个句子或一两段文字。

3.1.20

描述子组 descriptor group

一组相关的描述子。

3.1.21

可变点 variability point

制品中的一个点，当该制品所属的资产被应用于目标应用时，可在该点上进行修改或定制，以满足目标应用的个性化需求。

3.1.22

周境 context

一个框架性的引用或概念上的边界范围，为与其相关的事物确定含义。

3.1.23

包 package

组成资产的所有制品(文件)的集合。一个包可以实现为文件系统中的一个目录，或者一个存档文件。

3.1.24

根目录 root context

一个资产包的顶级目录，它定义了一个资产所有制品的边界(在允许以 URL 方式链接制品时除外)。

3.1.25

工具处理 tooling

用于描述对 RAS 实体描述文档和 RAS 资产包进行处理和管理的软件程序的一个通用术语。Rational XDE 就是一个可创建和使用 RAS 资产的商业化工具的实例。

3.2 缩略语

ABD	基于资产的开发(Asset-Based Development)
RAS	可复用资产规范(Reusable Asset Specification)
UML	统一建模语言(Uniform Modeling Language)
XML	可扩展置标语言(EXtensible Markup Language)
URL	统一资源定位符(Uniform Resource Locator)

4 约定

下列约定适用于本标准。

4.1 文档约定

文档约定如下：

a) 〈描述子-组〉元素：带有定界符〈 〉的术语代表一个 XML Schema 中的元素；

b) ***属性***：斜体加粗的术语是一个元素的属性；

c) 所有的节点和属性名称只能用小写字母表示；

d) 如果用多个单词作为一个节点或属性的名称，在字和字之间要使用连字符，比如“artifact-type”。

4.2 UML 建模约定

UML 建模约定如下：

a) **类名**——类名由大写字母开头，用连字号(“-”)连接多个独立的单词；

b) **关联、标识符、容器**——所有的关联被声明为传值(by-value)关联。这表示“被包含”的类是一个 XML Schema 中的子元素。因此,在需要保持持久性关联的地方,拥有者类要包含一个标识符(ID)属性;

c) **关联的基数**——类之间的基数使用 UML 的表示风格[下限…上限]来表示。在上限是无限的情况下,使用“*”表示;

d) **属性**——**属性名**与类名使用同样的规则,以小写字母开头,用连字号“-”连接多个独立的单词。
属性类型使用与编程语言无关的方式来声明,使用小写字母,例如[string,int]。
属性的必选/可选信息在属性的文档窗口中获取,值可以是[required,optional]。
属性的可见性在默认情况下被声明为私有的,但这些语义并不直接转换到 XML Schema 中。

5 可复用资产

5.1 导引

可复用软件资产(简称资产)提供了在一个给定的周境下对某问题的解决方案(solution)。图 1 是可复用资产的一个抽象描述。资产可有可变点,资产的消费者可在可变点上进行客户化。资产具有用于指导该资产如何使用的规则。

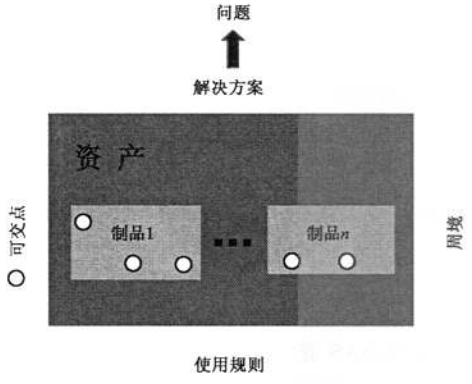

图 1 资产的基本概念

资产是一组制品的集合,制品可以是软件开发生存周期中的任何工作产品,例如需求文档、模型、源代码文件、部署描述子、测试用例或脚本等。

5.2 RAS 的基本模型

5.2.1 导引

一个资产蕴含了丰富的信息,不同类型的资产有不同的规约要求。本标准提供了规范各类资产的一个核心模型,在此称之为核心 RAS,并给出了基于核心 RAS 的扩展机制,基于核心 RAS 及其扩展机制来支持对特定类型资产的规约。本标准给出了针对两种资产类型的扩展:默认构件剖面和默认 Web Service 剖面(详见 5.4 和 5.5)。

本条主要阐述核心 RAS 及剖面扩展机制。

5.2.2 核心 RAS 模型及扩展机制

核心 RAS 是用于规范各类资产的一个核心模型,定义了用于规范各类资产的基本组成及其之间的关系。图 2 左边部分标识了核心 RAS 的一些主要部分,左图的顶端是一些资产级的属性。图 2 右边部分显示了核心 RAS 的主要组成部分之间的关系。核心 RAS 由 4 个主要部分组成:

a) 分类部分(Classification section),提供了一组用于资产分类的描述子以及与资产相关的周境描述;

b) 解决方案部分(Solution section),描述资产的制品;

c) 用法部分(Usage Section),包含安装、定制和使用资产的规则;

d) 相关资产部分(Related-Assets section),描述了与其他资产的关系。

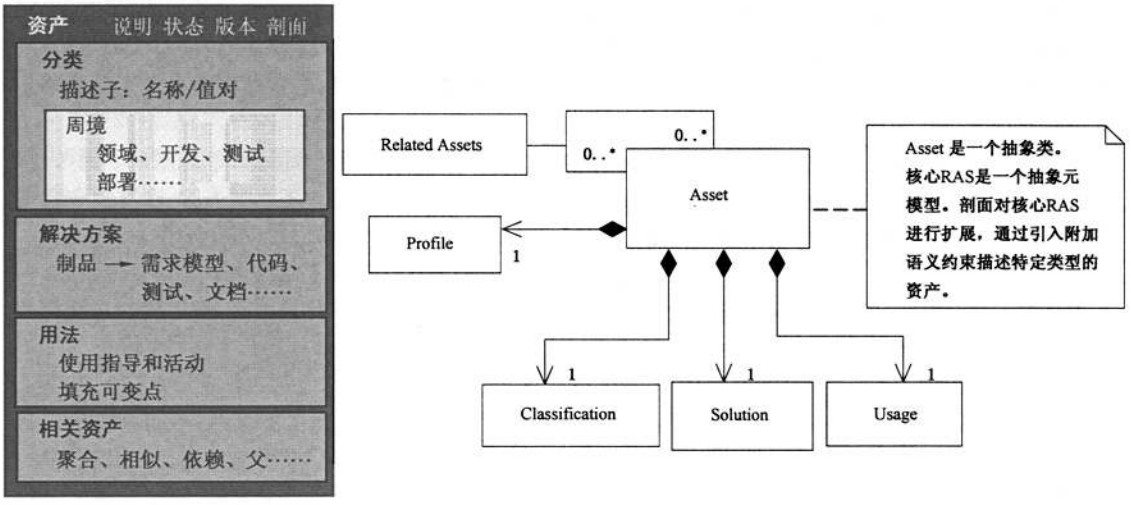

图 2　核心 RAS 的主要部分

本标准采用 UML 表示核心 RAS 模型，如图 3 所示。模型中的每个类代表一个资产元素，类间的聚合关系表示元素的拥有者和包含者之间的关系。关联关系描述了资产元素的关联，一般用标识符来保持关联关系。

对核心 RAS 各个类的详细阐述见 5.2.7～5.2.15。

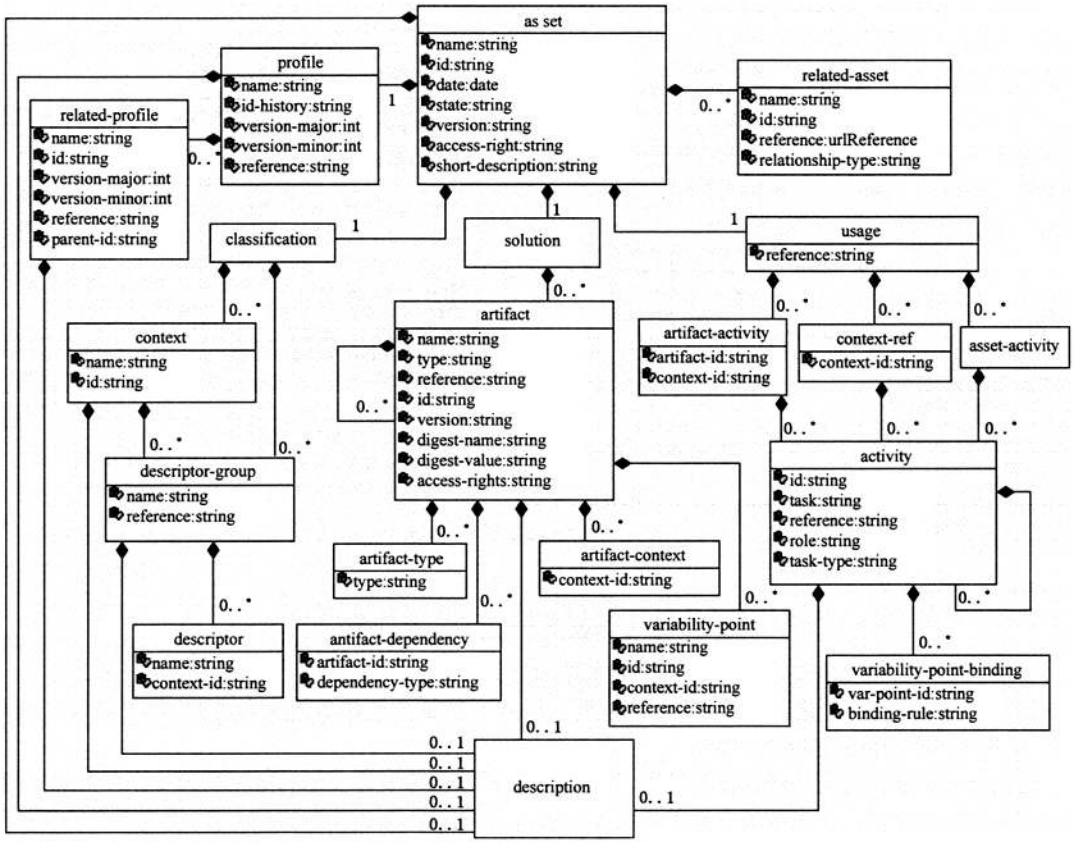

图 3　核心 RAS 的 UML 模型

通过对核心 RAS 进行扩展可得到某特定类型资产的描述模型。核心 RAS 模型采用与 UML 相同的扩展机制，这种扩展是增加或扩大基本模型信息的一种方法。一个描述模型可扩展核心 RAS 或其他描述模型，通过扩展可引入一些更为严格的语义和约束，但不能减少被扩展的描述模型中所规定的语义及约束。

可以在新扩展的描述模型中增加新的类，但当前描述模型中的类及其上的约束不能减少。例如，扩展得到的新描述模型可使当前模型的可选类成为必选类，但父模型中的约束不能被去除，现有类在新模型中的约束不能少于该类在父模型中的约束。

可以在新扩展的描述模型中增加当前类的属性，但当前类的属性及其上的约束不能减少。例如，一个新模型可使当前可选属性变为必选属性，且该属性在父模型中的约束不能被去除，现有属性在新模型中的约束不能少于该属性在父模型中的约束。

5.2.3 剖面 XML Schema 及扩展机制

核心 RAS 以及对其扩展得到的资产描述模型是一个抽象的模型，是非实例化的，不能直接用于资产的描述。剖面提供了资产描述模型的一个 XML Schema 实现，以核心 RAS 模型为例，图 3 模型中的每个类可用 XML Schema 元素来表示，根据该模型可派生出一个 XML Schema 文档，图 4 为该文档的主要部分，附加的语义约束见 5.2.15。该 XML Schema 连同附加的语义约束，提供了核心 RAS 的一个实现，即默认剖面。

采用核心 RAS 模型规范资产时，可通过默认剖面定义资产的实体描述文档结构、语法和语义约束，从而可具体描述一个特定的资产。

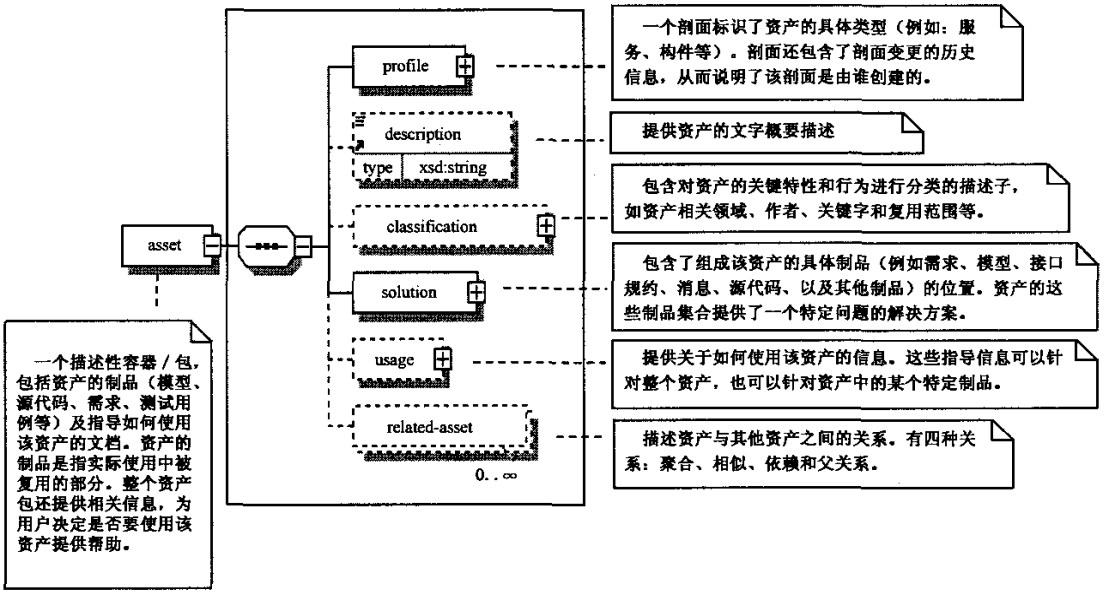

图 4 默认剖面的 XML Schema 概览

剖面采用与 UML 相同的扩展机制，这种扩展是增加或扩大基本(默认)规约信息的一种方法。一个剖面可扩展核心 RAS 或其他剖面，通过扩展可引入一些更为严格的语义和约束，但不能改变核心 RAS 或其他剖面所规定语义及约束。

可以在新扩展的剖面中增加新的元素，但当前剖面中的元素及其上的约束不能减少。例如，扩展得到的新剖面可使当前剖面的可选元素成为必选元素，但父剖面中的约束不能被去除，现有元素在新剖面中的约束不能少于该元素在父剖面中的约束。

可以在新扩展的剖面中增加当前元素的属性，但当前元素的属性及其上的约束不能减少。例如，一个新剖面可使当前可选属性变为必选属性，且该属性在父剖面中的约束不能被去除，现有属性在新剖面中的约束不能少于该属性在父剖面中的约束。

如图 5 所示，默认剖面是核心 RAS 的一个实现，默认构件剖面和默认 Web Service 剖面都派生自默认剖面。剖面派生信息可从剖面历史中得到，详见 5.2.9.1。

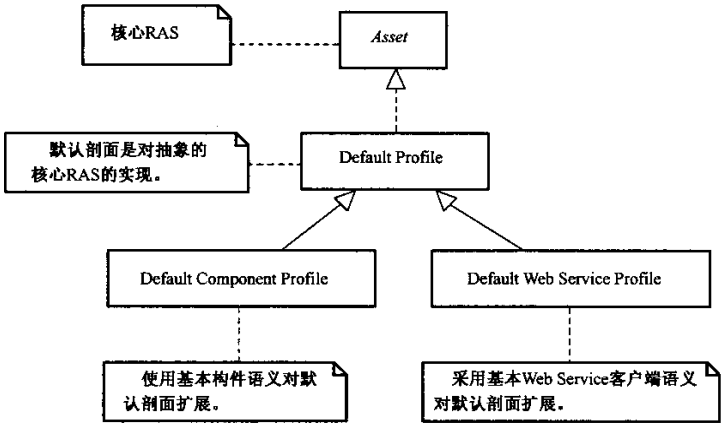

图 5　核心 RAS 与剖面扩展示例

本标准通过剖面扩展机制来支持对各类型资产（如构件、web services、模式和框架等）的描述。针对某特定资产类型，可通过对核心 RAS 或其他剖面的扩展得到一个适用于该类型资产描述的剖面，扩展后的剖面保留并扩展了核心 RAS 的语义及约束。剖面的扩展关系形成了剖面的谱系，默认剖面是所有具有 Xml schema 实现的剖面的祖先。

一个具体的剖面对应一个资产类型，用于对该类型中所有资产的描述，该剖面通过定义该类型资产实体描述文档的结构、语法和语义约束，来指导该类型中某特定资产的描述。一个按本标准打包的特定资产必须包含一个实体描述文档，该文档包含了资产的具体描述信息。剖面 XML Schema 文件一般与实体描述文档一起存放，也可采用 URL 引用方式，通过网络对其访问。

本标准中，核心 RAS 模型中的每个类对应了默认剖面的一个 XML Schema 元素，因此，在本标准后续的内容中，若无特别声明，对核心 RAS 模型中类的阐述同样适用于默认剖面中相应 XML Schema 元素的阐述，反之亦然。例如，对核心 RAS 中的 Asset 类的阐述，同样适用于默认剖面〈asset〉元素的阐述。

5.2.4　标准符合性

当以下条件都满足时，一个资产才认为是符合本标准的资产：

资产必须满足本标准中核心 RAS 所有的语义约束（见 5.2.15）。

5.2.5　必选类

本标准 5.2.15 的“语义约束”部分描述了满足标准符合性所必须赋值的类。其目的在于支持具体复用实施中形式化程度的差异。

可通过创建新的剖面以引入更为严格的语义和更多的必选类，已有的必选类在新的剖面中不能变为可选的（见 5.2.15 的“约束 15”）。

下面列出了必选类：

a)　Asset（资产）；

b)　Profile（剖面）；

c)　Solution（解决方案）。

尽管 Solution(解决方案)类是必选的,其相关的 Artifact(制品)类是可选的,但"约束 2"(见 5.2.15)规定,为满足标准符合性,必须至少存在一个有*name*(名称)或*reference*(引用)属性值的制品。将〈artifact〉作为可选元素是出于易于剖面将来变化的考虑,将来的剖面可能在解决方案部分增加更多的特定元素(例如需求,设计,实现等),可使这些元素是必选的,见 5.4 中对解决方案部分的扩展。

5.2.6 必选属性

只有少数属性是必选的。5.3.5 描述了为满足标准符合性,哪些附加属性必须予以赋值。其目的在于支持复用实施中形式化程度的差异。

可通过创建新的剖面以引入更为严格的语义和更多的必选属性,已有的必选属性在新的剖面中不能变为可选的(见 5.2.15 的"约束 15")。

表 1 列出了**必选属性**,这些属性中有许多存在于可选类中,意味着在打包资产时该类的 XML schema 节点不是必选的。

表 1 核心 RAS UML 模型的必选属性

必选类	必选属性		可选类	必选属性
Asset	name		Related-profile	name
	id			id
Profile	name			version-major
	id-history			version-minor
	version-major		Context	name
	version-minor			id
			Descriptor	name
			Artifact-context	context-id
			Artifact-dependency	artifact-id
			Variability-point	name
				id
			Artifact-type	type
			Artifact-activity	artifact-id
			Context-ref	context-id
			Activity	id
				task
			Variability-point-binding	variability-point-id
				binding-rule
			Related-asset	name
				relationship-type

5.2.7 Asset(资产)

每个实体描述文档都以单个 Asset 实例开始。Asset 实例定义了可复用软件资产的身份(见 5.2.14)。

Asset 实例包含两个必选的属性:*name* 和*id* 。属性*name*(资产名称)取值为简短字符串,该字符串应反映该资产的一般解决策略或资产可处理的问题,属性*id*(资产标识符)取值为全球唯一标识符,用于在工具化处理中区分不同的资产。

属性*date*(日期)包含一个符合默认 XML 格式(YYYY—MM—DD)的有效日期。该属性指明了

该资产可以被开始使用的时间。

属性*state*（状态）说明了资产当前的状态。该属性主要用于资产认证流程（该资产在资产库中发布之前所需经历的一系列评审）。

属性*version*（版本）是任意字符串，用于比较两个标识符属性相同的资产。

属性*access right*（访问权限）是任意字符串，用来说明资产消费者对资产的访问权限，例如查看或使用。

属性*short-description*（简短描述）为该资产的简短描述。用户查询资产库时一般最先看的信息是资产的*name*和*short-description*。

Asset类有两个必选的关联：Profile（剖面）和Solution（解决方案），还有4个可选的关联：Description（描述），Classification（分类），Usage（用法）和RelatedAsset（相关资产）。对这些类的讨论贯穿于整个标准。

5.2.8 Description（说明）

Description类是一个简单的容器，包含了关于该资产整体或某些组成元素的说明文字。Description在XML schema中是全局性的，并在多处被引用。例如〈asset〉元素下可以有〈description〉元素，其取值主要包括该资产处理的问题以及主要解决策略的较详细阐述；〈profile〉元素也可以有〈description〉元素用于说明当前资产剖面的有关信息；等等。

Description类的"值"在XML中用多行元素表示，如下所示：

〈description〉 The Description. value Here〈/description〉

5.2.9 Profile（剖面）

5.2.9.1 核心剖面

一个Profile定义了某一类型资产的实体描述文件的结构和语义。一个资产的实体描述文档必须标识一个用于验证自身的Profile类，该类精确定义了用于资产符合性所采用的特定剖面。Profile能引用其他模型元素（例如UML包或类）来对自身进行描述。Profile可能有不同的版本，应说明与其他剖面的谱系或祖先关系。RelatedProfile（相关剖面）从这个Profile的谱系关系中获取信息。

除了由核心RAS定义的、没有相应XML Schema实现的Core Profile（核心剖面），每一个Profile都是由其他剖面派生而来的。剖面可直接从核心RAS或其他任何一个剖面（例如默认剖面）进行扩展。剖面扩展只能在实体描述的XML Schema中增加元素和属性，以及/或者在已有元素上关联新的语义，而不能从XML Schema中去除元素或属性。一般来说，扩展后的剖面受更多限制。

属性*name*（剖面名）取值为一个可读的字符串，该字符串反映了当前剖面的目的或范围。属性*id*是剖面的权威标识符，一个剖面的*id*可能是任意字符序列，但该标识符在复用范围内必须是唯一的，且不能包含双冒号（::）。

属性*id-history*（标识历史）是一个组合键，由Profile的*id*及其所有祖先剖面的标识符串接而组成，串接符号为双冒号（::）。除了原始的Core Profile，一个剖面当且仅当派生自一个父剖面。

下面是默认剖面的*id-history*的示例：

F1C842AD-CE85-4261-ACA7-178C457018A1::31E5BFBF-B16E-4253-8037-98D70D07F35F

它说明了"F1C842AD-CE85-4261-ACA7-178C457018A1"标识的剖面是核心剖面，"31E5BFBF-B16E-4253-8037-98D70D07F35F"标识的是默认剖面。

如果定义了一个新的剖面，将产生一个新的id，应在剖面的标识历史中将其添加到它的父剖面标识符后面。这个新剖面对标识历史中所有在其之前的Profile进行了扩展。

属性*version-major*（主版本号）和*version-minor*（次版本号）取值均为整数，用于定义Profile的版本，特别是可用于区分与该Profile名称相同的以前的剖面。通常情况下，这两个值组合在一起形成浮

点数的形式。例如主版本号为2,次版本号为1,则可被写作版本2.1。当剖面被更新并且仍保留原有名称时,主版本号和次版本号的值应发生变化。例如,当对剖面进行更新但其用途和范围不变时,剖面名称应保持不变。

属性*reference*是一个可选属性,它引用一个外部文档来提供更多关于该剖面的信息,该文档应对剖面使用的新类、属性和语义进行解释。*reference*属性也可包含一个指向位于根目录之外资源的URL。

Profile类有两个关联,一个是与Description的关联,用于捕捉可读的、对该剖面的注释,另一个是与RelatedProfile的关联,用于提供该Profile的*id-history*中的每一个剖面的可读信息。

Profile可引用一个Artifact(制品)来为自身提供进一步的背景和解释。

5.2.9.2 RelatedProfile(相关剖面)

RelatedProfile类通过描述剖面的谱系来捕捉剖面的历史。〈relatedProfile〉元素的属性*name*,*id*,*version-major*,*version-minor*,*reference*,其含义与〈profile〉元素的属性相同,其中*id*属性包含了该剖面的标识符,它应在〈profile〉元素的*id-history*属性中出现。属性*parent-id*描述派生该剖面的父剖面。

RelatedProfile可引用一个Artifact,用于为RelatedProfile提供更进一步的背景和解释。

5.2.10 Classification(分类)

5.2.10.1 导引

在基于资产的开发中,搜索资产时采用的视角经常不同于其打包时的视角,因此资产的分类应支持多个视角。本标准支持以简单的名/值对方式进行分类,并允许用户引用自己定义的资产分类schema。

〈classification〉元素仅仅是一个实体描述中用于资产分类的所有元素的容器。〈classification〉元素没有定义任何属性。〈classification〉元素有两个可选的子元素:〈context〉和〈descriptor-group〉。

5.2.10.2 Context(周境)

周境定义了一个概念框架,用以解释资产中其他元素的含义。〈context〉元素定义了*name*(必选的)和*id*属性。实体描述中其他元素通过*id*引用周境。〈context〉元素有一个可选的〈description〉子元素,用于进一步说明周境的相关信息。一个〈context〉元素也可包含〈descriptor-group〉子元素,〈descriptor-group〉通过一组描述子(见5.2.10.4的〈descriptor〉)来服务于周境的定义。

实体描述文件中可定义多个周境。一个〈artifact〉可声明与多个周境的相关性。表2举例说明了周境的一些范畴的样本。本标准不规定周境的具体内容,下面的周境样本仅仅是示例。

表2 周境范畴

周境范畴	周境说明和举例
核心	与该周境相关的制品代表该资产的基本要素。若没有与该周境相关的制品或活动,资产就不能被成功应用于目标应用中。
业务	与该周境相关的制品是与一个具体的业务周境相关的。 例如:保险业,金融服务业
开发	与该周境相关的制品对于该资产的开发是必需的。这个周境通常出现在白盒资产中,用于可修改的制品。与该周境相关的制品可包括构建版脚本和工具,模型以及规约文档等。例如:J2EE 1.3,WebSphere Studio Application Developer 5.1等。
文档	与该周境相关的制品被用于资产的解释和文档化。这些制品并不是应用该资产所必需的。
运行	与该周境相关的制品对于资产的运行/执行是必需的。例如:WebSphere Application Server 5.1。
测试	与该周境相关的制品是与测试相关的,不需要直接应用到目标应用中。例如:脚本、样本数据或测试计划。

5.2.10.3 **DescriptorGroup(描述子组)**

DescriptorGroup 类仅仅是一组相关描述子的值的容器，描述子的值可能是描述子节点或自由形式的值。这些描述子的值可能来自一个或多个分类 schema，用于将一个具体资产相关的分类定义和值集中在一起。

属性*name* 为描述子组的名称，属性*reference* 指向另一个分类 schema 或本体。本标准不对行业和资产类型的所有可能进行阐述，属性*reference* 允许资产分类时采用多个分类 schema 或本体。

子元素〈description〉用于对描述子组提供进一步说明。

DescriptorGroup 可引用一个 Artifact，用于为 DescriptorGroup 提供进一步的背景和解释。

5.2.10.4 **Descriptor(描述子)**

该类定义了描述该资产质量和特性的分类描述子。属性*name* 为描述子名称，通常情况下，该名称在分类 schema 中是唯一的。属性*context-id* 包含一个周境标识符。

5.2.11 **Solution(解决方案)**

5.2.11.1 **导引**

一个资产提供一个解决方案，解决方案通过一个制品集合来提供。一个制品可包含其他制品或与其他制品相关联。一个制品可与一个特定的周境(如开发时或运行时周境)相关。一个制品可有可变点，供客户化定制。如图 6 所示。

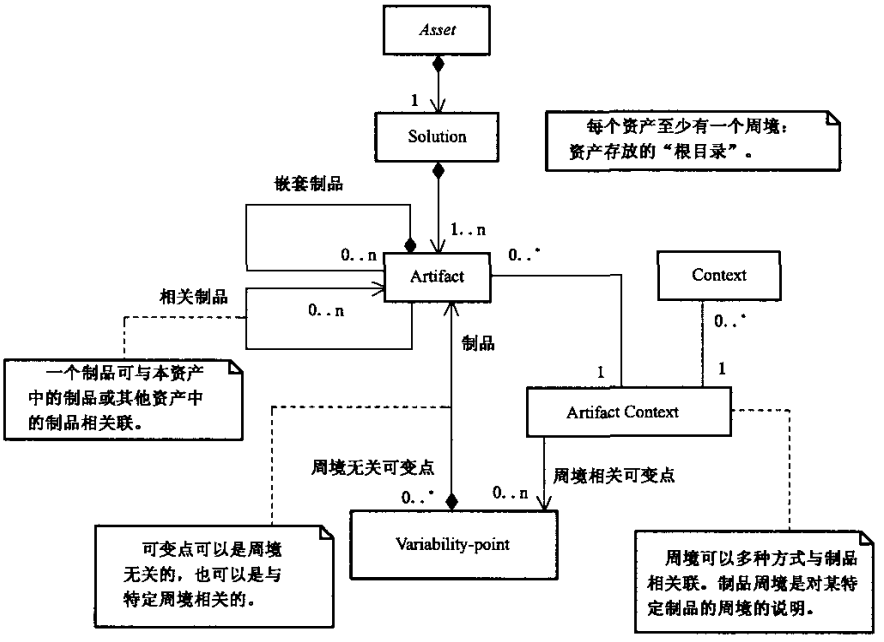

图 6 解决方案部分的领域模型

实体描述中的〈solution〉元素是一个包含该资产所有制品的容器。它是必选元素且没有指定任何属性。〈solution〉元素仅指定了〈artifact〉子元素。

5.2.11.2 **Artifact(制品)**

一个制品或者是位于资产包中的一个物理文件，或者代表一个逻辑实体，这个逻辑实体至少有一个子制品是物理文件。一个〈artifact〉元素必须至少指定一个*name* 或一个*reference* 属性。对代表逻辑实体的制品，其*name* 属性是必选的；对指定为(作为资产包的组成部分的)实际文件或工作产品的制品，

其*reference*属性是必选的。

属性*name*、*type*和*reference*是可选的，可通过工具批量添加具体名称未知的制品，*name*可以是制品的文件名，即允许与*reference*属性冗余。*reference*属性是可选的，从而允许〈artifact〉元素可包含其他〈artifact〉元素，并可引用文件系统或其他地方的制品。

一个〈artifact〉元素可指定一个供实体描述中其他元素所引用的*id*。*id*在实体描述文档中的所有制品范围内必须是唯一的。*version*属性(可选)用于标识一个制品的版本。

一个制品可采用某个特定算法进行加密。属性*digest-name*(加密名称)和*digest-value*(加密值)包含了加密的名称和值。

一个特定制品可指定与之相关的使用许可及权限。属性*access-rights*(访问权限)指定制品的许可信息，本标准不指定该许可信息的格式。

〈artifact〉元素可包含若干子元素：〈artifact-type〉、〈artifact-context〉、〈artifact-dependency〉、〈variability-point〉，同时也可包含子制品元素。

一个物理制品可以是任意类型的文件(二进制，纯文本等)。属性*type*指定制品的主类型的信息，该信息用于对制品的工具化处理。除主类型外，一个制品可指定任意数目的次类型。每个次类型由子元素〈artifact-type〉来详细说明(见5.2.11.6的〈artifact-type〉)。

主类型往往是制品文件的扩展名。例如，一个名为web.xml的文件，它的主类型是XML，次类型可能是J2EE Web Configure。

主类型列表将文件扩展名映射到类型名称。多个文件扩展名可对应于同一个类型，多个类型名称也可对应于同一个文件扩展名。在此情况下，应通过工具为用户处理两者间的映射提供便利。例如，一个名为usecases.doc的文件，其文件扩展.doc可映射为“Microsoft Word”类型和“WordPerfect”类型。

主类型(Primary Types)

下面列举的主类型样例来自于RASPrimaryArtifactTypes.xml文件：

```
〈artifact id = "dothtml"  type="Microsoft Word HTML Template"/〉
〈artifact id = "dox"  type="Visual Basic User Document Binary File"/〉
〈artifact id = "dqy"  type="Microsoft Excel ODBC Query files"/〉
〈artifact id = "drv"  type="Device driver"/〉
〈artifact id = "dsm"  type="DSM File"/〉
〈artifact id = "dsn"  type="Microsoft OLE DB Enumerator for ODBC Drivers"/〉
〈artifact id = "dsp"  type="Project File"/〉
〈artifact id = "dsr"  type="Visual Basic Designer Module"/〉
〈artifact id = "dtd"  type="Document Type Definition"/〉
〈artifact id = "dun"  type="Dialup Networking File"/〉
〈artifact id = "dv"  type="DV"/〉
〈artifact id = "DVD"  type=" DVD "/〉
〈artifact id = "ecs"  type=" Exchange Server Content Source "/〉
〈artifact id = "elm"  type=" Microsoft Office Themes File "/〉
〈artifact id = "eml"  type=" Internet E-Mail Message "/〉
〈artifact id = "ent"  type=" External Entity "/〉
〈artifact id = "enx"  type=" Rational XDE Unit "/〉
〈artifact id = "exc"  type=" Text "/〉
〈artifact id = "mdx"  type=" XDE Model "/〉
〈artifact id = "ifx"  type=" Rational XDE Unit "/〉
〈artifact id = "inx"  type=" Rational XDE Unit "/〉
```

这个列表是动态可变的，供工具生产商使用，以便他们可提供对制品的正确处理。

〈artifact-context〉元素允许一个制品与多个周境相关联。每个元素定义了一个单独的*context*。

〈artifact-dependency〉元素标识了当前制品与当前资产中其他制品的依赖。每一个元素包括一个

artifact-id（制品标识符）属性，它必须包含一个与当前实体描述中其他制品标识符值相同的值，该值不能自引用。*dependency-type*（依赖类型）属性用于描述依赖的类型，是可选的。

〈variability-point〉元素描述了制品中能被改变的内容及其位置。每一个〈variability-point〉指定了一个名称和一个标识符，实体描述中的其他元素可以通过该标识符引用可变点。可变点可与周境相关联，因此必须为文档中的〈context〉元素指定一个有效的标识符。*reference* 属性所指向的外部文档对〈variability-point〉作了进一步的解释说明。

一个制品可包含若干子制品。子制品使用相同的〈artifact〉元素。如果制品是一个逻辑制品，则必须指定一个*name* 且至少有一个子制品是物理文件。

5.2.11.3 ArtifactContext(制品周境)

〈artifact-context〉元素将一个周境和一个制品相关联。见 5.2.11.2 的〈artifact〉元素说明。

5.2.11.4 ArtifactDependency(制品依赖)

ArtifactDependency 标识一个依赖的 Artifact。这个依赖的 Artifact 必须是实体描述中定义的其他 Artifact，见 5.2.11.2 的 Artifact 说明。ArtifactDependency 是 Artifact 类的一个子类。属性*dependencyType*（依赖类型）描述了制品依赖的类型，例如设计时、编译时或运行时的依赖。

5.2.11.5 VariabilityPoint(可变点)

每一个〈variability-point〉标识了在具体使用该制品时一个可供修改的位置。其*name* 和*id* 属性是必选的。*id* 供实体描述中的其他元素引用(见 5.2.12.6 的〈variability-point-binding〉元素说明)。可选情况下，〈variability-point〉可通过周境的*context-id*（周境标识符）属性建立与一个周境的关联。〈variability-point〉元素的自由形式文本(应为纯文本方式)用于对该活动(可变点绑定)的更为完整的说明。可选的*reference* 属性指向一个外部文档，该文档可对可变点进行更进一步的解释说明。

VariabilityPoint 可引用一个 Artifact，用于为 VariabilityPoint 提供更进一步的背景和阐释。

VariabilityPoint 可包含一个对自身提供附加注释的 Description。

5.2.11.6 ArtifactType(制品类型)

〈artifact〉可由多个类型来描述。在此描述了两种制品类型：主类型和次类型。主类型用于工具处理时当前制品进行主要的动作和操作，次类型主要用于说明的目的和工具处理中的次要动作。

〈artifact〉元素的*type* 属性用于表示主类型。每个〈artifact〉有且只有一个主类型，但可以有多个次类型。〈artifact-type〉元素的*type* 属性用于说明次类型。

次类型

如果应用了错误的主类型，引入资产时工具可能不会对该类型进行任何专门处理。工具提供商必须提供对主类型列表的处理(见 5.2.15 的“约束 12”)，对于次类型列表的处理则不是必需的。

次类型列表包括了主类型列表，并增加了一些主要用于说明目的的新类型。下面是 RASSeondaryArtifactTypes. xml 文件中的次类型列表的样例：

```
〈artifact id="usecase" type="Use Case"/〉
〈artifact id="testcase" type="Test Case"/〉
〈artifact id="reqmodel" type="Analysis Model"/〉
〈artifact id="designmodel" type="Design Model"/〉
〈artifact id="implmodel" type="Implementation Model"/〉
〈artifact id="testmodel" type="Test Model"/〉
〈artifact id="busncncptmodel" type="Business Concept Model"/〉
〈artifact id="usecasemodel" type="Use Case Model"/〉
〈artifact id="busntypemodel" type="Business Type Model"/〉
〈artifact id="intfacespecmodel" type="Interface Spec Model"/〉
〈artifact id="websvcintermodel" type="Web Service Interactions Model"/〉
〈artifact id="analysisset " type="Analysis Artifact Set"/〉
〈artifact id="designset " type="Design Artifact Set"/〉
〈artifact id="implset " type="Implementation Artifact Set"/〉
〈artifact id="testset " type="Test Artifact Set"/〉
〈artifact id="implset " type="Implementation Artifact Set"/〉
```

```
〈artifact id="testset " type="Test Artifact Set"/〉
〈artifact id="intfacespecdiag" type="Interface Spec Diagram"/〉
〈artifact id="usecasediag" type="Use Case Diagram"/〉
〈artifact id="compinterdiag" type="Component Interaction Diagram"/〉
```

以上列表是动态可变的，工具生产商应根据该列表对制品提供正确的处理。

5.2.12 Usage(用法)

5.2.12.1 导引

用法部分描述了应用资产时应执行的活动，一般采用轻载的活动或工作流模型来描述。指导资产如何使用的活动有几种形式：一些活动适用于整个资产，另一些活动适用于资产的某个制品，还有些活动与特定的周境相关。特定周境下的一个制品可包含相关的可变点。如图 7 所示。

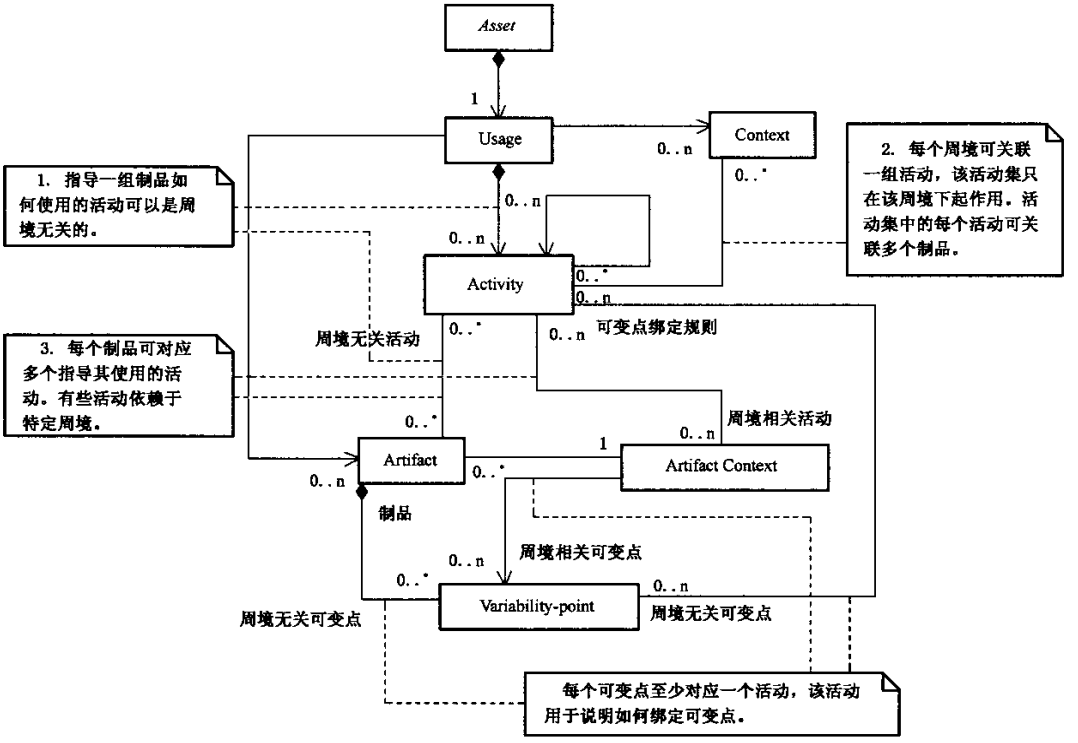

图 7 用法部分的领域模型

〈usage〉元素是一个容器元素，用于对过程或用法的指导。该元素只定义了一个属性*reference*，它指向一个外部文档，这个文档可以是对用法部分的总体解释，或是对该资产所有使用活动的总结。

用法元素有三类子元素，每一类都包含一组将资产应用到目标应用时应遵循的活动。每一个子元素可能有多个实例，这意味着可能有多个〈artifact-activity〉、〈context-ref〉和〈asset-activity〉子元素。

Usage 可引用一个 Artifact，用于为 Usage 提供更进一步的背景和解释。

5.2.12.2 ArtifactActivity(制品活动)

〈artifact-activity〉元素是一个容器，包括一组与特定制品相关的活动。它包括两个属性：*artifact-id*（制品标识符）和*context-id*（周境标识符）。*artifact-id* 属性是必选的，它指定了一个实体描述文档中某制品的标识符。一个制品可有多个活动，活动可与周境相关联。属性*context-id* 的值必须是实体

描述文档中的某个〈context〉元素的标识符。

〈artifact-activity〉至少应包含一个〈activity〉元素，它指定了将资产应用到目标应用中时，用户或工具应执行的具体活动。见 5.2.11.2 的〈artifact〉元素说明。

5.2.12.3 ContextRef(周境引用)

〈context-ref〉元素是一个与特定周境相关联的活动的容器。***context-id*** 属性是必选的，它指定了一个在实体描述文档中定义的周境。

〈context-ref〉至少应包含一个〈activity〉元素，它指定了将资产应用到目标应用中时，用户或工具应执行的具体活动。见 5.2.11.2 的〈artifact〉元素说明。

5.2.12.4 AssetActivity(资产活动)

〈asset-activity〉元素是一个与整个资产相关联的活动的容器。资产活动元素没有定义任何属性。

5.2.12.5 Activity(活动)

活动指定了使用资产时用户或工具对该资产进行的具体操作。〈activity〉元素包括两个必选属性：***id*** 和***task***（任务）。属性***id*** 应包含一个标识符，用于支持工具处理，该标识符在实体描述文件的所有活动中是唯一的。属性***task*** 以简短说明或关键字的形式描述活动的意图或目标。对活动更详尽的说明可通过〈description〉子元素或***reference*** 属性（可选）所指向的外部文档来提供。***reference*** 属性也可指向一个供工具使用的、包含可执行代码或脚本的文件。

一些活动可能与某些资产消费者角色相关，因此***role*** 属性声明了该活动与哪种资产消费者角色相关。***task-type***（任务类型）属性对活动类型进行划分。该属性可供工具使用，并说明了如何执行当前活动。一些任务类型可提示工具载入并执行一段脚本，或运行一个可执行程序，而另一些任务类型则可能仅仅指示该活动应在消费者的 To Do List(操作列表)中被引用。

除了〈description〉元素外，〈activity〉元素可包含子〈activity〉元素和〈variability-point-binding〉元素。〈variability-point-binding 〉是对制品中定义的可变点的引用，并规定了绑定规则。绑定规则是关于可变点和活动之间关系的简短说明。

Activity 可引用一个 Artifact，用于为 Activity 提供更进一步的背景和解释。

5.2.12.6 VariabilityPointBinding(可变点绑定)

〈variability-point-binding〉有两个属性，属性***var-point-id*** 指定了一个实体描述文件中定义的可变点标识符，属性***binding-rule***（绑定规则）是可变点特性的简短描述，该属性提供了将〈activity〉作用于〈variability-point〉时资产消费者应遵循的额外规则和指示说明。

下面是一个部分文法的样例：

```
〈solution〉
      〈artifact〉 name：Design Model，id ：100，reference ：model/designmodel.mdx
          〈variability-point〉 name ：Design Model::User Account Management::Use Case_Create New User
          Account ，id ：1
〈usage〉
      〈artifact-activity〉artifact-id ：100
           〈activity〉 id ：5，task ：Specify the alternate flows
               〈variability-point-binding〉 variability-point-id ：1，binding-rule ：Provide a description of
          invalid database connection flow.
                〈variability-point-binding〉 variability-point-id：1，binding-rule：Do not create new rela-
                tionships to existing packages.
```

在这个例子中，〈variability-point〉标识了〈artifact〉中实施客户化的具体位置。〈activity〉标识了资产消费者实施客户化时的应该做什么，〈variability-point-binding〉描述了资产消费者实施客户化时的规则、约束和附加的指导。

5.2.13 RelatedAsset(相关资产)

资产通常与其他资产存在某种关系,资产的关系信息将有助于减少复用成本。一个资产可拥有任意数目的相关资产。每个相关资产通过一个〈related-asset〉元素来规约。一个相关资产可以是当前资产范围之外的资产。

当资产复用扩大至大粒度或者粗粒度级别时,〈related-asset〉元素是必选的,在该级别上,以资产族或资产集的形式来定义或复用资产。

属性*name* 包含相关资产的名称。

属性*relationship-type* (关系类型)可以是任意值,但对于一些特定的关系类型本标准定义了相应的保留值,这些关系类型以及对应的保留值如下:

a) 聚合(aggregation):表示当前资产"包含"了相关资产,该包含关系可通过传值或引用来实现;

b) 相似(similar):表示其他资产有与当前资产相似的特性;

c) 依赖(dependency):表示当前资产引用或依赖于该相关资产的某些服务或制品;

d) 父(parent):表示该相关资产包含或拥有当前资产。

包含多个资产的资产包可使用聚合关系和父关系类型来组织被包含的资产。

属性*asset-id* 指定了该相关资产的资产标识符。

属性*reference* 指定了该相关资产的位置。可以采用 URL 或文件路径方式指定资产位置,也可以是对相关资产的一个说明文档的引用。

属性*assetVersion* (资产版本)指明了该相关资产的版本。

RelatedAsset 可引用一个 Artifact,用于为 RelatedAsset 提供更进一步的背景和解释。

RelatedAsset 可包含 Description,用于对 RelatedAsset 提供额外注释。

5.2.14 AssetIdentity(资产身份)

某个特定的资产可通过资产标识符和版本号唯一确定。一旦资产标识符改变,它就成为一个全新的资产。当资产演变为一个全新的资产时,建议在新资产的实体描述中的〈related-asset〉部分引用原资产。

在资产标识符(id)不变的情况下,通过版本号标识资产的演化关系,资产的版本是一个任意的字符串。建议用一个统一的数字标号系统来表示版本,标号系统应便于对名称/标识符相同的资产进行版本字符串的比较,便于区分版本的新旧,在新版本中,资产的名称、简短说明或其他元数据信息都可能会改变。

5.2.15 核心 RAS 语义约束

语义约束是关于资产实体描述中不能通过标准 XML Schemas 表示的规则。以下的约束加上剖面的 XML Schema,完整定义了一个有效的实体描述文件。

约束 1　实体描述文件必须是经剖面 XML Schema 验证有效的。

约束 2　在资产的解决方案元素内必须至少有一个引用属性和*name* 属性均为非空的制品元素。

约束 3　一个资产中的一个文件最多只能与一个〈artifact〉元素相关联。

约束 4　〈artifact-context〉、〈descriptor〉、〈artifact-dependency〉、〈variability-point〉、〈context-ref〉和〈artifact-activity〉元素的*context-id* 属性指定的周境元素必须出现在同一个实体描述文档中。

约束 5　〈artifact-activity〉和〈artifact-dependency〉元素中的*artifact-id* 属性指定的〈artifact〉元素必须出现在同一个实体描述文档中。

约束 6　〈variability-point-binding〉元素的 variability-point-id 属性指定的〈variability-point〉元素必须出现在同一个实体描述文档中。

约束 7　如果〈related-asset〉元素的*asset-id* 属性被赋值，则该值必须是其他实体描述文档中的〈asset〉元素的*id* 属性值。

约束 8　〈related-asset〉元素的*relationship-type* 属性允许为任何值，本标准规定的保留关系和取值见 5.2.13 的“相关资产”。

约束 9　〈profile〉元素的*id-history*（标识符历史）属性必须包含一个由其全部祖先剖面标识符首尾相连而形成的串。这些剖面标识符必须用 2 个连续的冒号来分隔。在*id-history* 中，越靠左的剖面标识符在世系关系中辈分越高，越靠右则辈分越低。

剖面没有指定标识符值的类型。但在该资产剖面的预期复用范围内，标识符值必须是唯一的。

约束 10　实体描述文件不能在〈artifact〉元素中引用它自身，否则会引起资产信息和元信息的混乱。

约束 11　〈artifact-dependency〉和〈artifact-activity〉元素的*artifact-id* 属性的值必须是同一个文档中〈artifact〉元素的 id。

约束 12　〈artifact〉元素的*type* 属性值必须使用主类型值。次类型值必须通过〈artifact-type〉元素来处理。

约束 13　每个〈artifact〉元素是（被称为资产根目录的）〈context〉元素的一部分。但该周境是隐含的，不需要在每个〈artifact〉中指定。

约束 14　可以通过创建一个剖面来引入更严格的语义和约束。例如，可使当前剖面中的可选类在新剖面中成为必选类。但是在父剖面中的约束不能在子剖面中被移除。例如，已有类在新剖面中的约束不能少于该类在父剖面中的约束。

约束 15　可在新的剖面中增加现有类的属性。然而，已有的属性约束不能减少。例如，可使当前剖面中的可选属性在新剖面中成为必选的。但是在父剖面中的约束不能在子剖面中被移除。已有属性在新剖面中的约束不能少于该属性在父剖面中的约束。

5.3 默认剖面

5.3.1 导引

默认剖面是核心 RAS 的一个实现。核心 RAS 和默认剖面各自作为一个单独的实体维护，核心 RAS 未对其实现中的实施细节进行规范。用户定制（客户化）的剖面应扩展自默认剖面或者本标准中其他的剖面，如默认 Web Service 剖面或者默认构件剖面。

5.3.2 新元素概要

默认剖面反映了前面章节中描述的核心 RAS。该默认剖面的 XML Schema 见附录 B。

5.3.3 必选类

该剖面使用核心 RAS 规范的所有必选类，没有增加新的类。

5.3.4 必选属性

该剖面使用核心 RAS 规范的所有必选属性，没有增加新的属性。

5.3.5 语义约束

该剖面没有额外的语义约束。核心 RAS 的语义约束均应用于该剖面。

5.3.6 标准符合性

当以下所有条件都满足时，基于该剖面的资产才认为是符合本标准的：

核心 RAS 的标准符合性被保持。见 5.2.4。

5.4 默认构件剖面

5.4.1 导引

默认构件剖面派生自默认剖面。这里构件的语义吸取了参考文献[1]中对构件原理和概念的阐述，默认构件剖面支持参考文献[1]中规范构件所需的一些模型和图表。

5.4.2 必选类

“语义约束”(见 5.3.5)描述了某些元素的规则。除了默认剖面中必选的类之外，默认构件剖面增加了下列必选类：

a) Operation(操作)

5.4.3 必选属性

除了默认剖面中的必选属性外，默认构件剖面增加了以下必选属性：

表 3 默认构件剖面的 UML 模型的必选属性

必选类	必选属性		可选类	必选属性
Operation	name		Association-role	name
	initiates-transaction			type
			Attribute	name
				type
			Condition	description
				type
			Diagram-dependency	diagram-id
			Interface-spec	name
			Model-dependency	model-id
			Parameter	direction
				name
				type

5.4.4 标准符合性

当以下所有条件都满足时，基于该剖面的资产是符合本标准的：

a) 默认剖面的标准符合性被保持。见 5.3.6 的“标准符合性”；

b) 默认构件剖面的约束被保持。见 5.4.6 的“默认构件剖面语义约束”。

5.4.5 Solution(解决方案)

5.4.5.1 导引

这里只阐述默认构件剖面中新增的类，其他元素的信息可参考默认剖面。解决方案部分新增了 4 个类，包括：Requirement(需求)、Design(设计)、Implementation(实现)和 Test(测试)。这些元素对特殊类型的 Artifacts(制品)进行组织，以改进对资产的浏览和导航。

本部分的模型仅显示了默认构件剖面中新增的类。默认构件剖面对 Solution 类进行了扩展，图 8 阐明了 Solution 部分的类及其之间的关系。

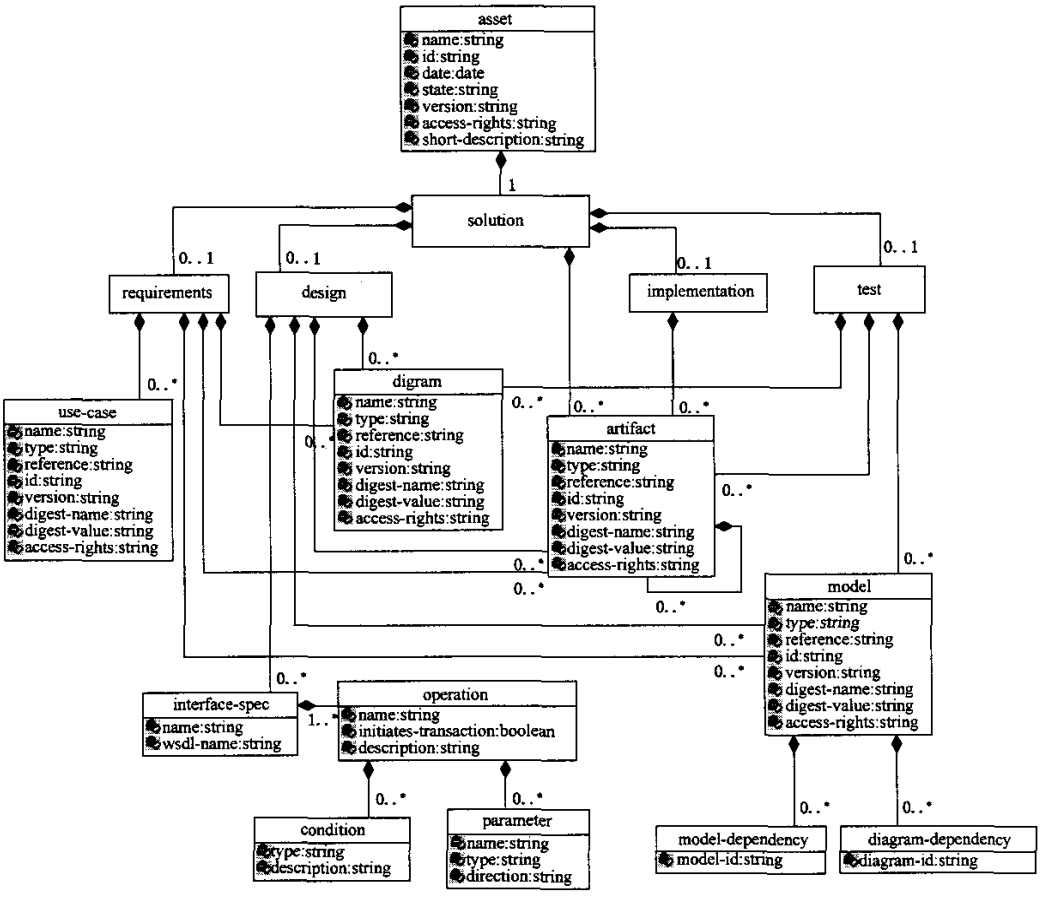

图 8　默认构件剖面的 UML 模型

5.4.5.2　Requirements(需求)

Requirements 类没有指定任何属性，但它与 Model(模型)、Diagram(图)、UseCase(用况)和 Artifact 类相关联。〈requirements〉元素内的模型、图、制品等用于描述该构件预期实现的需求。模型、图和制品节点在 XML schema 中是全局性的。

5.4.5.3　Model(模型)

Model 类对构件预期实现的需求模型进行规范，Modle 类的属性与制品的属性相同。可以有多个模型，如商业概念模型以及用况模型。

5.4.5.4　DiagramDependency(图依赖)

DiagramDependency 类用于在模型与图之间建立关系，便于消费者进行资产浏览与评价时对某特定模型的所有图的理解。

diagram-id 属性应引用当前实体描述文档中的一个 Diagram。

5.4.5.5　ModelDependency(模型依赖)

ModelDependency 类用于建立模型之间的关系。可通过一系列模型来指导资产消费者对构件的理解。

该类的*model-id*属性包含了同一实体描述文件中的 Model 的标识符值。

5.4.5.6 **Diagram(图)**

一个模型可以有多个图。对于每个 Requirements、Design 以及 Test 类,Diagram 类标识了相关的图,例如商业概念模型图以及用况图。

5.4.5.7 **UseCase(用况)**

构件可实现一个或多个用况,该类指向一个用况描述。

该类的属性与 Artifact 的属性相同,但被限制为对构件的用况文档的引用。

5.4.5.8 **Design(设计)**

Design 类没有属性,但它与 Model、Diagram、InterfaceSpec(接口规约)和 Artifact 类相关联。这些模型、图和制品等用于描述资产消费者使用该构件所需的设计元素。

5.4.5.9 **InterfaceSpec(接口规约)**

InterfaceSpec 类描述构件的一个接口。一个构件可定义多个接口,因而该类可能需要多个实例。属性*name*是用户使用的构件接口名称。属性*description*是对该接口的简短描述。该类与 Operation 类关联。如果要创建一个 InterfaceSpec 的实例,则必须创建一个或多个 Operation。

InterfaceSpec 可包含一个对其提供附加注释的 Description 类。

5.4.5.10 **Operation(操作)**

Operation 类描述了一个接口操作,并与 Condition(条件)及 Parameter(参数)两个类相关联。这两个类在资产打包中提供了足够的信息,以使资产消费者及工具可对接口的特性进行推理。

Operation 类有 3 个属性,*name*、*initiates-transaction* 以及*description*。*name* 是该操作的名称。*initiates-transaction*(布尔量)声明该 Operation 是否启动了一个事务。*description* 提供了对 Operation 的一个简要抽象。

Operation 可包含一个对其提供附加注释的 Description 类。

5.4.5.11 **Condition(条件)**

Condition 类捕获当前操作的前置、后置和其他条件。它有两个属性:*type* 指明条件的类型,*description* 提供对当前条件的解释。

Condition 可包含一个对其提供附加注释的 Description 类。

5.4.5.12 **Parameter(参数)**

Parameter 类使用*name*、*type* 以及*direction* 属性来描述 Operation 上的参数。*name* 属性是该参数的名称。*type* 属性描述了参数的类型。*direction* 属性描述了该参数是当前操作的输入参数、输出参数、或者两者都是。

5.4.5.13 **Implementation(实施)**

Implementation 类是一组 Artifacts(制品)的集合。这些 Artifact(制品)标识了提供构件实施的二进制及其他形式的文件。Implementation 类没有指定任何属性。

5.4.5.14 **Test(测试)**

Test 类没有指定任何属性,它有几个关联类:Model、Diagram 和 Artifact。〈test〉元素中的模型、图、制品等用于描述资产消费者对构件的测试。模型、图和制品元素在 XML schema 中是全局性的。

5.4.5.15 **默认构件剖面的语义约束**

约束 1:对于〈requirements〉元素;其子元素宜只包含与需求相关的制品。见 5.4.5.2。

对于〈design〉元素;其子元素宜只包含与设计相关的制品。见 5.4.5.8。

对于〈implementation〉元素;其子元素宜只包含与实现相关的制品。见 5.4.5.13。

对于〈test〉元素;其子元素宜只包含与测试相关的制品。见 5.4.5.14。

所有其他制品宜在〈solution〉元素的子元素〈artifact〉中处理。见 5.4.5。

约束 2：〈diagram-dependency〉中的***diagram-id*** 属性宜引用当前实体描述文档中〈diagram〉元素的 id。见 5.4.5.4 和 5.4.5.6。

约束 3：〈model-dependency〉元素的***model-id*** 属性包含一个来自于同一实体描述文档中的〈model〉元素的标识符值。见 5.4.5.3 和 5.4.5.5。

约束 4：一旦创建了〈interface-spec〉元素，必须创建一个或多个相应的〈operation〉元素。见 5.4.5.9 和 5.4.5.10。

约束 5：〈condition〉元素的***type*** 属性应赋以“pre”(前置)，“post”(后置)等值。见 5.4.5.11。

约束 6：〈parameter〉元素的***direction*** 属性宜赋以“in”(输入)，“out”(输出)，“inout”(输入/出)值。见 5.4.5.12。

5.5 默认 Web Service 剖面

5.5.1 导引

Web services 提供了拥有操作、参数和保证状态的信息模型的若干接口。其特性本质上与构件相类似，然而其部署和实例化模型有着明显的不同。

默认 Web Service 剖面派生自默认剖面，它描述了一个 Web Service 的客户端部分，它与构件在编程模型上的存在着许多相似之处。

5.5.2 必选类

语义约束(见 5.3.5)部分描述了某些元素的规则。除了在默认剖面中的必选的类外，默认 Web Service 剖面增加了以下必选类：

a) Impelementation；

b) Wsdl。

5.5.3 必选属性

除了在默认剖面中的必选属性外，默认 Web Service 剖面增加了以下必选属性：

表 4 默认 Web Service 剖面的 UML 模型的必选属性

必选类	必选属性		可选类	必选属性
Implementation	—		Interface-spec	wsdl-name
Wsdl	reference			

5.5.4 标准符合性

当以下所有条件都满足时，基于该剖面的资产是符合本标准的：

a) 默认剖面的标准符合性被保持，见 5.3.6；

b) 默认 Web Service 剖面的约束被保持。见 5.5.6。

5.5.5 解决方案(Solution)

5.5.5.1 导引

这里只阐述默认 Web Service 剖面中新增的类，其他类的信息可参考该剖面的祖先，即默认剖面。

本部分的模型仅显示了默认 Web Service 剖面中新增的类。默认 Web Service 剖面对 Solution 类进行了扩展，图 9 阐明了 Solution 部分的类及其之间的关系。

Solution(解决方案)部分新增了 4 类，包括：Requirement(需求)、Design(设计)、Implementation(实现)和 Test(测试)。这些类对特殊种类的 Artifacts(制品)进行组织，以改进对资产的浏览和导航，并规范了一些必选的 WSDL 元素。

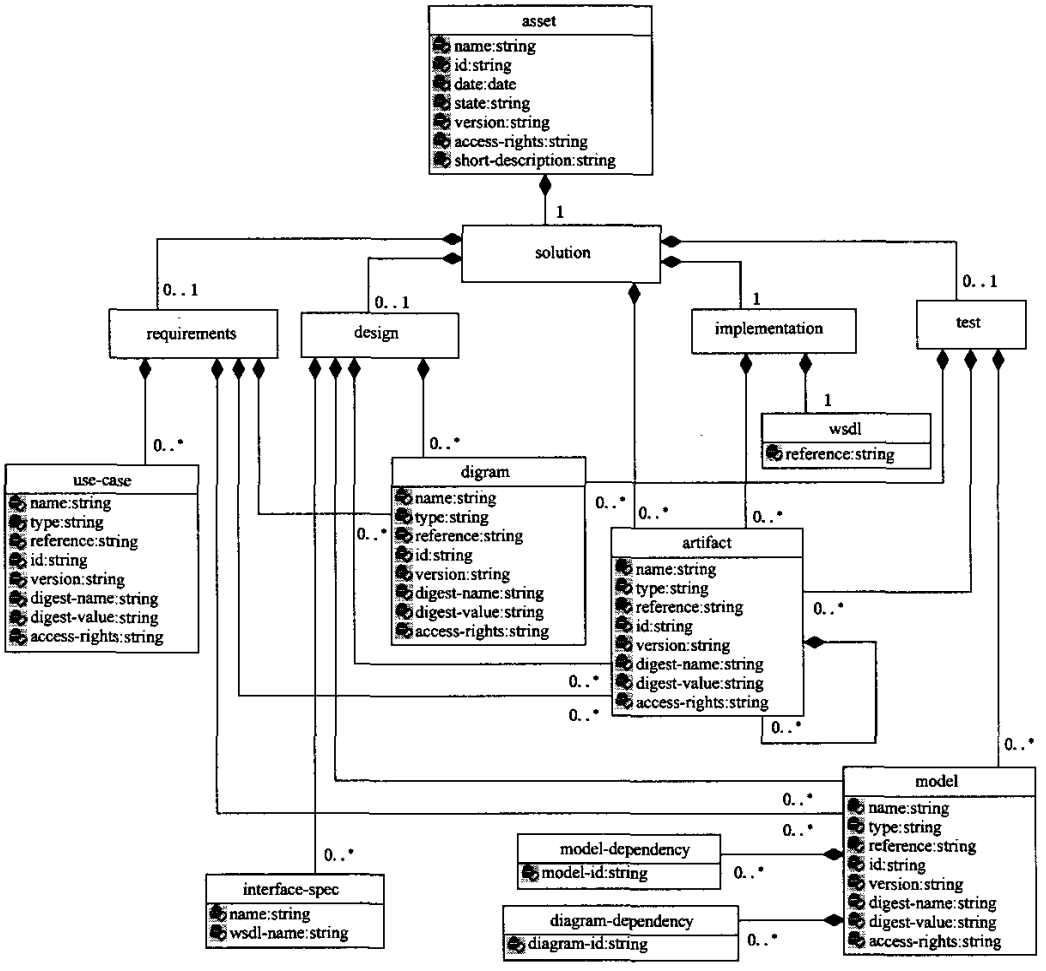

图 9　默认 Web Service 剖面的 UML 模型

5.5.5.2　InterfaceSpec(接口规约)

InterfaceSpec 类指向 wsdl 文件中的描述接口设计的部分。Web service 可定义多个接口，因而可能需要有多个该类的实例。属性*name* 是供用户使用的接口名称，*wsdl-name* 是在 wsdl 中使用的名称。Wsdl 类提供了对该接口上若干操作的形式化描述的一个引用。

5.5.5.3　Implementation(实施)

Implementation 类是必需的，它没有指定任何属性。Implementation 类有一组 Artifacts，这些 Artifacts 标识了提供 web service 实施的二进制和其他形式的文件。Implementation 类与 Wsdl 类相关联。

5.5.5.4　Wsdl

Wsdl 类引用了包含 web service 描述的文件。

5.5.6　默认 Web Service 剖面语义约束

约束 1：对于〈requirements〉元素，其子元素宜只包含与需求相关的制品。见 5.5.5。

对于〈design〉元素；其子元素宜只包含与设计相关的制品。见 5.5.5。

对于〈implementation〉元素；其子元素宜只包含与实现相关的制品。见 5.5.5.3。

对于〈test〉元素；其子元素宜只包含与测试相关的制品。见 5.5.5。

所有其他制品宜在〈solution〉元素的子元素〈artifact〉中进行处理。见 5.5.5。

约束 2：〈diagram-dependency〉中的***diagram-id*** 属性宜引用当前实体描述文档中〈diagram〉元素的 id。见 5.5.5。

约束 3：〈model-dependency〉元素的 model-id 属性包含一个来自于同一实体描述文档中的〈model〉元素的标识符值。见 5.5.5。

附 录 A
（资料性附录）
资产的打包

A.1 资产的打包方式

每个有效的可复用资产必须包含一个实体描述文件（下面将说明）和至少一个制品。该实体描述文件是一个 XML 文档，该文档是对某已知 RAS XML Schema（见实体描述 Schema）验证有效的文档，并满足相应的剖面文档描述中的附加语义约束。

一个资产包包括一个制品文件集合和一个实体描述。资产打包有以下几种方式：

a） 捆绑性打包。打包成一个存档文件；

b） 非捆绑性打包：

1） 制品可保持它原来的位置；

2） 在“打包”时制品可移动到其他位置。

A.1.1 捆绑性打包

该打包方法可用于一个团队开发环境，它采用压缩方法将包括实体描述文件在内的所有文件打包进一个档案文件中，这就使资产分发更为方便。该资产打包方法由图 A.1 说明。

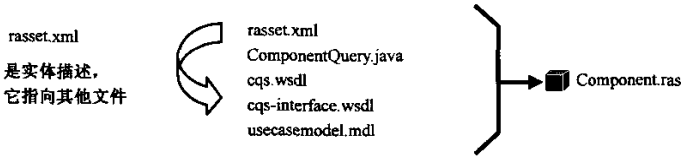

图 A.1 采用压缩方法进行资产打包

实体描述文件所在（在文件系统或在一个档案文件中）的目录被认为是该资产的全部制品（文件）的根目录。实体描述文件中引用的所有文件的路径都是相对于根目录而言的。当资产以压缩方法打包时，实体描述中引用的所有文件都应存在于根目录或其子目录中。实体描述中对文件的引用是相对于根目录的引用。

对于将其制品打包在. ras 文件中的资产，将“rasset. xml”作为实体描述的文件名。该文件应放在. ras 文件的根目录下。. ras 文件中可有多个. rmd 文件，但必须有且只有一个位于. ras 文件的根目录下的. rmd 文件，该文件是该资产的入口访问点。

对一个资产包中所包含的附加文件没有限制。资产包中的附加文件可以是出于工具处理或过程中提高实用或实效性的原因而存在。然而这些附加文件不应被视为资产的一部分。所有组成这个资产的文件，以及使用该资产时所需的文件，必须被资产实体描述的〈solution〉元素中的一个且仅有一个〈artifact〉元素引用。未被〈artifact〉元素所引用的文件被视为非该资产所需的（额外）文件，如果去除这些额外文件，用工具对该资产重新打包和发布修订后的资产包是完全合法的，且该重新打包的资产包将被视为与原资产包等价。有两种文件不受上述约束限制（即必须被〈artifact〉元素引用），第一种是资产实体描述文件（即 rasset. xml），第二种是 RAS XML Schema 文件。

对于聚集的资产或定义了多个资产的包，所有实体描述文件中的所有〈artifact〉节点的引用的全集决定了在转移或复制资产时哪些文件需要保留在该包中。

A.1.2 非捆绑性打包（制品在原位置）

在团队环境中开发制品通常要使用版本控制系统。定义资产的一个方法是将资产实体描述文件加入版本控制系统，并指向制品的原有位置。RAS 结构支持这种风格的资产“打包”。

A.1.3 非捆绑性打包(制品移动到新位置处)

在许多情况下当制品被打包进一个资产时,制品需要一些改动以使它们可复用。从这点来说,制品可以一个新的身份呈现,并且被移动到新的位置以便进行必要的修改。同样的,RAS 结构支持这种情况的资产"打包"。

A.2 .ras 文件格式

A.2.1 映射 RAS 到.ras 文件

RAS 仅仅是一个书面的规约,需要更形式化的表示以支持工具处理,为此采用 XML Schema(即.xsd 文件)来描述。为创建 XML Schema,在此采用 RAS UML 模型作为生成初始.xsd 文件的基线。该文件的第一个版本是可描述任何类型资产的默认剖面。

基于默认剖面的 XML Schema,可以创建包含某特定资产内容描述所需的元素的 XML 文档,该 XML 文档称为实体描述文件,名称为 rasset.xml,该文档是资产的访问入口点。

实体描述文件位于资产的"根"目录。该文件伴有一个.xsd(XML Schema)文件以及任意其他的制品、文件、子目录等。这些文件被压缩为一个扩展名为.ras 的文件。图 A.2 说明了 RAS 和.ras 文件的关系。

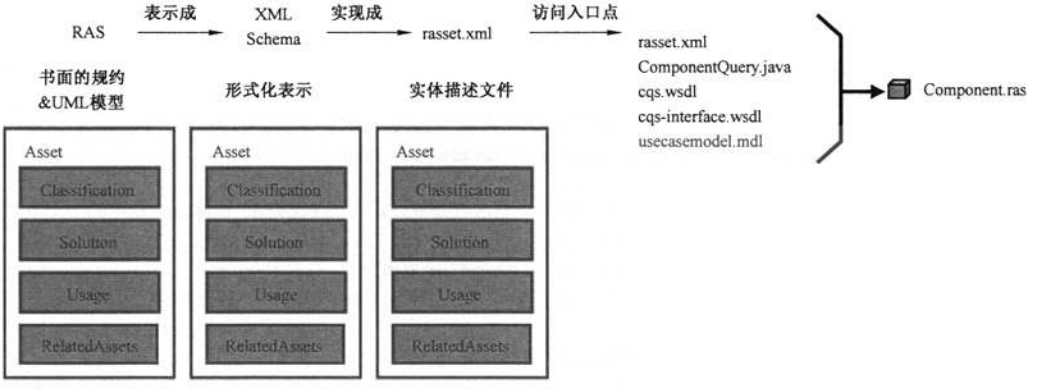

图 A.2 映射 RAS 到.ras 文件

每个.ras 文件包含以下类型的文件:

a) 零或多个 XML Schema 文件(如:RASProfile.xsd);

b) 位于根目录中的一个实体描述文件(如 rasset.xml),也可有其他的实体描述文件;

c) 一个或多个制品文件(如:源代码、模型、测试脚本等)。

.ras 文件的每个制品(rasset.xml 文件和 XML Schema 文件除外)必须在 rasset.xml 的〈solution〉元素中被引用。每个制品(即:文件)应在.ras 文件中出现一次。

A.2.1.1 组织.ras 文件

.ras 文件可在文件系统上组织,或由版本控制系统来组织。可按资产类型、版本、或状态等来组织。

A.2.1.2 浏览.ras 文件

工具提供商在对资产进行列表显示时,可检查.ras 文件中的 rasset.xml 文件以提取资产名称和简短描述。rasset.xml 文件结构可方便地转换为易于浏览的 HTML。

附 录 B
（规范性附录）
默认剖面的 XML Schema

```
<? xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:defaultprofile="http:///defaultprofile.ecore"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace=
    "http:///defaultprofile.ecore">
    <xsd:complexType name="Classification">
        <xsd:sequence>
            <xsd:element name="descriptorGroup" type="defaultprofile:DescriptorGroup"
minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="context" type="defaultprofile:Context" minOccurs="0" maxOccurs="unbound-
ed"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="Classification" type="defaultprofile:Classification">
    </xsd:element>
    <xsd:complexType name="Asset">
        <xsd:sequence>
            <xsd:element name="classification" type="defaultprofile:Classification" minOccurs="0"/>
            <xsd:element name="solution" type="defaultprofile:Solution"/>
            <xsd:element name="usage" type="defaultprofile:Usage" minOccurs="0"/>
            <xsd:element name="relatedAsset" type="defaultprofile:RelatedAsset" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="profile" type="defaultprofile:Profile"/>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="id" type="xsd:string"/>
      </xsd:attribute>
        <xsd:attribute name="date" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="state" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="version" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="accessRights" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="shortDescription" type="xsd:string">
        </xsd:attribute>
    </xsd:complexType>
    <xsd:element name="Asset" type="defaultprofile:Asset">
    </xsd:element>
    <xsd:complexType name="Solution">
        <xsd:sequence>
            <xsd:element name="artifact" type="defaultprofile:Artifact" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="Solution" type="defaultprofile:Solution">
    </xsd:element>
    <xsd:complexType name="Artifact">
```

```
        〈xsd:sequence〉
            〈xsd:element name="artifactContext" type="defaultprofile:ArtifactContext" minOccurs="0"
maxOccurs="unbounded"/〉
            〈xsd:element name="artifactDependency" type="defaultprofile:ArtifactDependency"
minOccurs="0" maxOccurs="unbounded"/〉
            〈xsd:element name="artifact" type="defaultprofile:Artifact" minOccurs="0"
maxOccurs="unbounded"/〉
            〈xsd:element name="variabilityPoint" type="defaultprofile:VariabilityPoint" minOccurs="0"
maxOccurs="unbounded"/〉
            〈xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/〉
            〈xsd:element name="artifactType" type="defaultprofile:ArtifactType" minOccurs="0"
maxOccurs="unbounded"/〉
            〈xsd:element name="reference" type="defaultprofile:Reference" minOccurs="0"/〉
        〈/xsd:sequence〉
        〈xsd:attribute name="name" type="xsd:string"〉
        〈/xsd:attribute〉
        〈xsd:attribute name="type" type="xsd:string"〉
        〈/xsd:attribute〉
        〈xsd:attribute name="version" type="xsd:string"〉
        〈/xsd:attribute〉
        〈xsd:attribute name="digestName" type="xsd:string"〉
        〈/xsd:attribute〉
        〈xsd:attribute name="digestValue" type="xsd:string"〉
        〈/xsd:attribute〉
        〈xsd:attribute name="accessRights" type="xsd:string"〉
        〈/xsd:attribute〉
    〈/xsd:complexType〉
    〈xsd:element name="Artifact" type="defaultprofile:Artifact"〉
    〈/xsd:element〉
    〈xsd:complexType name="Usage"〉
        〈xsd:sequence〉
            〈xsd:element name="contextRef" type="defaultprofile:ContextRef" minOccurs="0"
maxOccurs="unbounded"/〉
            〈xsd:element name="artifactActivity" type="defaultprofile:ArtifactActivity" minOccurs="0"
maxOccurs="unbounded"/〉
            〈xsd:element name="assetActivity" type="defaultprofile:AssetActivity" minOccurs="0"
maxOccurs="unbounded"/〉
        〈/xsd:sequence〉
        〈xsd:attribute name="artifact" type="xsd:string"〉
            〈xsd:annotation〉
                〈xsd:documentation〉{optional}〈/xsd:documentation〉
            〈/xsd:annotation〉
        〈/xsd:attribute〉
    〈/xsd:complexType〉
    〈xsd:element name="Usage" type="defaultprofile:Usage"〉
        〈xsd:annotation〉
            〈xsd:documentation〉xmi:ordered=false〈/xsd:documentation〉
        〈/xsd:annotation〉
    〈/xsd:element〉
    〈xsd:complexType name="RelatedAsset"〉
        〈xsd:annotation〉
            〈xsd:documentation〉xmi:contentType=mixed〈/xsd:documentation〉
        〈/xsd:annotation〉
        〈xsd:sequence〉
            〈xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/〉
        〈/xsd:sequence〉
```

```
        〈xsd:attribute name="name" type="xsd:string"/〉
        〈xsd:attribute name="relationshipType" type="xsd:string"〉
            〈xsd:annotation〉
                〈xsd:documentation〉OPTIONAL〈/xsd:documentation〉
            〈/xsd:annotation〉
        〈/xsd:attribute〉
        〈xsd:attribute name="assetId" type="xsd:string"/〉
        〈xsd:attribute name="assetVersion" type="xsd:string"/〉
        〈xsd:attribute name="artifact" type="xsd:string"/〉
    〈/xsd:complexType〉
    〈xsd:element name="RelatedAsset" type="defaultprofile:RelatedAsset"〉
        〈xsd:annotation〉
            〈xsd:documentation〉xmi:contentType=mixed〈/xsd:documentation〉
        〈/xsd:annotation〉
    〈/xsd:element〉
    〈xsd:complexType name="Context"〉
        〈xsd:annotation〉
            〈xsd:documentation〉xmi:ordered=true 〈/xsd:documentation〉
        〈/xsd:annotation〉
        〈xsd:sequence〉
            〈xsd:element name="descriptorGroup" type="defaultprofile:DescriptorGroup" minOccurs="0"
maxOccurs="unbounded"/〉
            〈xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/〉
        〈/xsd:sequence〉
        〈xsd:attribute name="name" type="xsd:string"〉
            〈xsd:annotation〉
                〈xsd:documentation〉{required}〈/xsd:documentation〉
            〈/xsd:annotation〉
        〈/xsd:attribute〉
        〈xsd:attribute name="id" type="xsd:string"/〉
        〈/xsd:attribute〉
    〈/xsd:complexType〉
    〈xsd:element name="Context" type="defaultprofile:Context"〉
        〈xsd:annotation〉
            〈xsd:documentation〉xmi:ordered=true〈/xsd:documentation〉
        〈/xsd:annotation〉
    〈/xsd:element〉
    〈xsd:complexType name="ContextRef"〉
        〈xsd:annotation〉
            〈xsd:documentation〉xmi:ordered=true〈/xsd:documentation〉
        〈/xsd:annotation〉
        〈xsd:sequence〉
            〈xsd:element name="activity" type="defaultprofile:Activity" minOccurs="0"
maxOccurs="unbounded"/〉
        〈/xsd:sequence〉
        〈xsd:attribute name="context" type="xsd:string"/〉
    〈/xsd:complexType〉
    〈xsd:element name="ContextRef" type="defaultprofile:ContextRef"〉
        〈xsd:annotation〉
            〈xsd:documentation〉xmi:ordered=true〈/xsd:documentation〉
        〈/xsd:annotation〉
    〈/xsd:element〉
    〈xsd:complexType name="DescriptorGroup"〉
        〈xsd:sequence〉
            〈xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/〉
            〈xsd:element name="contained" type="defaultprofile:DescriptorGroup" minOccurs="0"
```

```
maxOccurs="unbounded"/>
            <xsd:element name="freeFormValue" type="defaultprofile:FreeFormValue" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="freeFormDescriptor" type="defaultprofile:FreeFormDescriptor"
minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="artifact" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="nodeDescriptor" type="xsd:string">
            <xsd:annotation>
                <xsd:documentation>ordered</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="classificationSchema" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="DescriptorGroup" type="defaultprofile:DescriptorGroup">
    </xsd:element>
    <xsd:complexType name="FreeFormDescriptor">
        <xsd:complexContent>
            <xsd:extension base="defaultprofile:Descriptor"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="FreeFormDescriptor" type="defaultprofile:FreeFormDescriptor">
    </xsd:element>
    <xsd:complexType name="ArtifactContext">
        <xsd:annotation>
            <xsd:documentation>An ArtifactContext&gt; element associates a Context to an
Artifact.</xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="context" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="ArtifactContext" type="defaultprofile:ArtifactContext">
        <xsd:annotation>
            <xsd:documentation>An ArtifactContext&gt; element associates a Context to an
Artifact.</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
    <xsd:complexType name="ArtifactDependency">
        <xsd:sequence>
            <xsd:element name="dependencyKind" type="defaultprofile:DependencyKind"/>
        </xsd:sequence>
        <xsd:attribute name="dependencyType" type="xsd:string"/>
        <xsd:attribute name="artifact" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="ArtifactDependency" type="defaultprofile:ArtifactDependency">
    </xsd:element>
    <xsd:complexType name="ArtifactActivity">
        <xsd:annotation>
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="activity" type="defaultprofile:Activity" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
```

```
        <xsd:attribute name="context" type="xsd:string"/>
        <xsd:attribute name="artifact" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="ArtifactActivity" type="defaultprofile:ArtifactActivity">
        <xsd:annotation>
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
    <xsd:complexType name="AssetActivity">
        <xsd:annotation>
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="activity" type="defaultprofile:Activity" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="AssetActivity" type="defaultprofile:AssetActivity">
        <xsd:annotation>
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
    <xsd:complexType name="Activity">
        <xsd:annotation>
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="activity" type="defaultprofile:Activity" minOccurs="0"
maxOccurs="unbounded"/>
             <xsd:element name="variabilityPointBinding" type="defaultprofile:VariabilityPointBinding"
minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
            <xsd:element name="activityParameter" type="defaultprofile:ActivityParameter" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="task" type="xsd:string">
            <xsd:annotation>
                <xsd:documentation>{required}</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
<xsd:attribute name="id" type="xsd:string"/>
</xsd:attribute>
        <xsd:attribute name="role" type="xsd:string">
            <xsd:annotation>
                <xsd:documentation>{optional}</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="taskType" type="xsd:string"/>
        <xsd:attribute name="artifact" type="xsd:string">
            <xsd:annotation>
                <xsd:documentation>{optional}</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:element name="Activity" type="defaultprofile:Activity">
        <xsd:annotation>
```

```
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
    <xsd:complexType name="VariabilityPoint">
        <xsd:sequence>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string">
            <xsd:annotation>
                <xsd:documentation>{required}</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
       <xsd:attribute name="id" type="xsd:string"/>
        <xsd:attribute name="context" type="xsd:string"/>
        <xsd:attribute name="artifact" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="VariabilityPoint" type="defaultprofile:VariabilityPoint">
    </xsd:element>
    <xsd:complexType name="VariabilityPointBinding">
        <xsd:attribute name="bindingRule" type="xsd:string">
            <xsd:annotation>
                <xsd:documentation>{required}</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="variabilityPoint" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="VariabilityPointBinding" type="defaultprofile:VariabilityPointBinding"/>
    <xsd:complexType name="Profile">
        <xsd:annotation>
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
             <xsd:element name="relatedProfile" type="defaultprofile:RelatedProfile" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="history" type="defaultprofile:Description" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="id" type="xsd:string"/>
        </xsd:attribute>
        <xsd:attribute name="idHistory" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="versionMajor" type="xsd:int">
        </xsd:attribute>
        <xsd:attribute name="versionMinor" type="xsd:int">
        </xsd:attribute>
        <xsd:attribute name="artifact" type="xsd:string"/>
        <xsd:attribute name="element" type="xsd:string"/>
        <xsd:attribute name="classificationSchema" type="xsd:string"/>
        <xsd:attribute name="dependencyKind" type="xsd:string"/>
        <xsd:attribute name="requiredElement" type="xsd:string"/>
        <xsd:attribute name="requiredAttribute" type="xsd:string"/>
        <xsd:attribute name="semanticConstraint" type="xsd:string"/>
    </xsd:complexType>
```

```
    <xsd:element name="Profile" type="defaultprofile:Profile">
        <xsd:annotation>
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
    <xsd:complexType name="RelatedProfile">
        <xsd:annotation>
            <xsd:documentation>xmi:ordered=true</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="id" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="versionMajor" type="xsd:int">
        </xsd:attribute>
        <xsd:attribute name="versionMinor" type="xsd:int">
        </xsd:attribute>
        <xsd:attribute name="parentId" type="xsd:string">
        </xsd:attribute>
        <xsd:attribute name="artifact" type="xsd:string">
        </xsd:attribute>
    </xsd:complexType>
    <xsd:element name="RelatedProfile" type="defaultprofile:RelatedProfile">
    </xsd:element>
    <xsd:complexType name="ArtifactType">
        <xsd:attribute name="type" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="ArtifactType" type="defaultprofile:ArtifactType"/>
    <xsd:complexType name="DependencyKind">
        <xsd:attribute name="name" type="xsd:string"/>
        <xsd:attribute name="profile" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="DependencyKind" type="defaultprofile:DependencyKind"/>
    <xsd:complexType name="Description">
        <xsd:attribute name="value" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="Description" type="defaultprofile:Description">
    </xsd:element>
    <xsd:complexType name="NodeDescriptor">
        <xsd:complexContent>
            <xsd:extension base="defaultprofile:Descriptor">
                <xsd:sequence>
                    <xsd:element name="specific" type="defaultprofile:NodeDescriptor" minOccurs="0"
maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="exclusive" type="xsd:boolean" default="False"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="NodeDescriptor" type="defaultprofile:NodeDescriptor"/>
    <xsd:complexType name="ClassificationSchema">
        <xsd:sequence>
            <xsd:element name="descriptor" type="defaultprofile:Descriptor" minOccurs="0"
```

```
maxOccurs="unbounded"/>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string"/>
        <xsd:attribute name="profile" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="ClassificationSchema" type="defaultprofile:ClassificationSchema"/>
    <xsd:complexType name="Descriptor">
        <xsd:sequence>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string"/>
    </xsd:complexType>
    <xsd:complexType name="Reference">
        <xsd:sequence>
            <xsd:element name="referenceKind" type="defaultprofile:ReferenceKind"/>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:string"/>
        <xsd:attribute name="element" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="Reference" type="defaultprofile:Reference"/>
    <xsd:complexType name="ReferenceKind">
        <xsd:sequence>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="ReferenceKind" type="defaultprofile:ReferenceKind"/>
    <xsd:complexType name="ActivityParameter">
        <xsd:sequence>
            <xsd:element name="description" type="defaultprofile:Description" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="defaultValue" type="xsd:string"/>
        <xsd:attribute name="name" type="xsd:string"/>
        <xsd:attribute name="value" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="ActivityParameter" type="defaultprofile:ActivityParameter"/>
    <xsd:complexType name="FreeFormValue">
        <xsd:attribute name="value" type="xsd:string"/>
        <xsd:attribute name="freeFormDescriptor" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="FreeFormValue" type="defaultprofile:FreeFormValue"/>
</xsd:schema>
```

参 考 文 献

[1] John Cheesman, John Daniels. UML Components: A Simple Process for Specifying Component-Based Software, Addison Wesley, October 2000.