

ICS 35. 240. 01

A11

备案号:

JR

中华人民共和国金融行业标准

JR/T 0107.2—2014/ISO 17369-2: 2013

统计数据 and 元数据交换 (SDMX) 第 2 部分: 信息模型 UML 概念设计

Statistical data and metadata exchange (SDMX) —
Part 2: Information model: UML conceptual design

(ISO 17369-2:2013, Information model: UML conceptual design, IDT)

2014 – 08 – 28 发布

2014 – 08 – 28 实施

中国人民银行

发布

目 次

前言	III
引言	IV
1 范围	1
2 术语和定义	1
3 介绍	2
3.1 建模技术和图解	2
3.2 总体功能性	2
4 执行者和用例	3
4.1 执行者和用例	4
4.2 用例图	4
5 SDMX 基础包	10
5.1 引言	10
5.2 标识、版本和维护	10
5.3 数据类型	13
5.4 项目方案模式	16
5.5 结构模式	18
6 具体项目方案	25
6.1 引言	25
6.2 继承视图	25
6.3 代码表	25
6.4 概念方案	27
6.5 类别方案	30
6.6 组织方案	33
6.7 项目方案关联	36
7 数据结构定义和数据集	38
7.1 介绍	38
7.2 继承视图	38
7.3 数据结构定义——关联视图	40
7.4 数据集——相关性视图	49
8 立方体	57
8.1 背景	57
8.2 在信息模型中对 cube 的支持	57
9 元数据结构定义和元数据集	57

9.1 背景	57
9.2 继承	58
9.3 元数据结构定义	60
9.4 元数据集	65
10 层级代码方案	70
10.1 范围	70
10.2 继承	71
10.3 关系	72
11 结构集和映射	74
11.1 范围	74
11.2 结构集	74
11.3 结构映射	77
11.4 项目方案映射	80
11.5 混合代码表映射	83
12 约束条件	85
12.1 范围	85
12.2 继承	85
12.3 约束条件	86
13 数据提供	94
13.1 类图	94
13.2 图解	95
14 进程	98
14.1 介绍	98
14.2 模型-继承和关系视角	98
15 转换和表达式	100
15.1 范围	100
15.2 模型-继承视角	100
附录 A（资料性附录）SDMX 信息模型中关于 UML 的简要指南	104
参考文献	110

前 言

JR/T 0107《统计数据和元数据交换（SDMX）》分为七个部分：

- 第1部分：框架；
- 第2部分：信息模型 UML 概念设计；
- 第3部分：SDMX-ML 模式和文档；
- 第4部分：SDMX-EDI 语法和文档；
- 第5部分：注册表规范 逻辑功能和逻辑接口；
- 第6部分：SDMX 技术说明事项；
- 第7部分：Web 服务用法指南。

本部分为JR/T 0107的第2部分。

本部分按照 GB/T 1.1-2009 规则起草。

本部分等同采用 ISO 17369-2:2013《统计数据和元数据交换（SDMX） 第2部分：信息模型 UML 概念设计》。

本部分由中国人民银行提出。

本部分由全国金融标准化技术委员会（SAC/TC180）归口。

本部分起草单位：中国人民银行调查统计司、中国金融电子化公司。

本部分主要起草人：盛松成、徐诺金、姚力、巴运红、任全忠、潘润红、李曙光、韩建国、贾树辉、李兴锋、吴隼、廖燕平、王媛、司燕翔、刘蔚、刘运、邓琳莹、李静。

引 言

统计数据和元数据交换（SDMX）标准由 SDMX 国际组织发起并提出。SDMX 国际组织是由国际清算银行（BIS）、经济合作与发展组织（OECD）、欧盟统计局（Eurostat）、欧洲中央银行（ECB）、国际货币基金组织（IMF）、联合国（UN）和世界银行（WB）七个国际组织联合建立，其制定发布的《统计数据和元数据交换》标准规定了统计人员在采集、处理和交换统计数据时所使用的统计概念和方法，规范了对外披露统计信息时统计数据的机构范围、地理区域、存流量性质、时间属性、频度以及文件格式等内容。

SDMX 标准提供了统计数据及元数据交换和共享的标准化格式，可以达到更好地扩展和高效率使用的目的。目前 SDMX 标准主要应用领域为部分国家中央银行和统计部门。本部分的作用在于规范我国金融统计标准体系的内部处理和对外发布，促进金融统计的互联互通、信息共享和业务协同，提高信息共享的效率，满足金融综合统计的需要。

统计数据和元数据交换(SDMX)

第2部分：信息模型 UML 概念设计

1 范围

本部分规定了SDMX信息模型，包括SDMX基础包、具体项目方案、关键字族、立方体和元数据结构定义和元数据集、层级代码方案、结构集和映射、数据约束和供应等。附录A中给出了UML指南以及针对不熟悉用关键字族的方法描述统计数据结构的人员的指南。

本部分适用于金融统计中数据和元数据的交换和共享。

2 术语和定义

本标准第1部分的以及下列术语和定义适用于本文件。

2.1

数据流定义 dataflow definition

标识了数据结构定义并可能与分类中一个或多个专题域有关（该内容有助于根据分类模式查找数据）。

注：在SDMX中，根据数据流定义报告或发布数据集。根据报告周期性或数据集中允许的可能关键字内容的子集，约束条件可附加给数据流定义。

2.2

元数据流定义 metadataflow definition

与数据流定义非常类似，但其描述、分类并限制的是元数据集。

2.3

提供协议 provision agreement

描述了数据供应方提供数据集和元数据集方式的信息集合。

注：可以与数据或元数据流定义非常类似的方式对供应协议进行限制。因此，数据供应方可表述为其提供了一个覆盖一系列特定国家和主题的特殊数据流。重要的一点是，已注册数据或元数据的实际来源附加于供应协议中（以URL形式给出）。使用术语“协议”的原因是该信息可作为“服务等级协议”的基础被理解。然而，在SDMX中，和任何类合同信息不同（该内容不在本技术规范范围内）的是，其以信息元数据支持技术系统。

2.4

报告分类 reporting taxonomy

允许组织链接（可能以分级方式）一系列立方体或数据流定义，这些立方体或数据流定义组合构成完整的数据或元数据“报告”。

注：该报告分类支持基本的报告，通常这些报告构成了异构数据的复合立方体，也可支持其他收集和报告功能，以及通过出版物中的数据或元数据支持出版规范（如年鉴）。

3 介绍

3.1 建模技术和图解

用于SDMX信息模型（SDMX-IM）的建模技术是统一建模语言（UML），附录A给出了SDMX-IM所用UML结构的概述。

UML图允许显示类时带有属性和操作（有时称作方法）中的一个或两个。本部分中，因为没有操作，所以不显示操作（见图1、图2）。

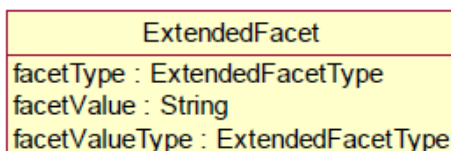


图1 隐藏操作的类

在有些图中，尽管一些类可能有属性，但会隐藏，这样做是为了图的清晰。使用的原则是：
 ——定义了类且它的属性和关联也定义了时，属性将一直出现在类图表上；
 ——在其它图表上，如继承图表，为了清晰度，可能隐藏类的属性。

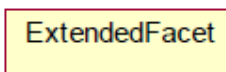


图2 隐藏操作和属性的类

注意，在任何情况下，从超类中继承的属性都不在子类中显示。
 表1的结构用于类、属性、关联的定义中。

表1 定义

类	特征	说明
ClassName		
	attributeName	
	associationName	
	+roleName	

“特征”栏的内容包括或解释了下面的类的结构特征之一：

- 是否是一个抽象类。抽象类用斜体 Courier 字体显示；
- 该类的超类（如果有）；
- 该类的子类（如果有）；
- Attribute: 用 Courier 字体显示；
- Association: 用 Courier 字体显示。如果关联是来自于超类之间的关联，则格式是 /associationName；
- Role: +roleName 用 Courier 字体显示。

“说明”栏提供对类或特征的简短定义或解释。可能在描述中使用UML类名，如果这样，它们用正常字体，且两个字之间有空格，例如类CodeList应写成Code List。

3.2 总体功能性

3.2.1 信息模型包

SDMX信息模型（SDMX-IM）是一个概念化的元模型，从该模型中开发了语法的具体实现。该模型由一组功能包构成，这种构造形式有助于理解、重用和维护模型。

另外，为了帮助理解，每个包都可以被认为是在三个概念层之一中：

- SDMX 基础层（SDMX Base layer）由结构定义层（Structural Definitions layer）和报告和分发层（Reporting and Dissemination layer）使用的基本块组成；
- 结构定义层由所需的支持数据和元数据报告和分发的结构化工具的定义组成；
- 报告和分发层由用于报告和分发的数据和元数据容器的定义组成。

实际上，层没有隐式或显式的结构功能，因为任意包能使用其它包里的任何部分。

3.2.2 本部分的三层结构

本部分主要支持用立方体（cube）结构类型的知识系统进行数据分析，如OLAP系统（On line analytical processing），其中包括：

- 元数据结构定义；
- 元数据集；
- 层级代码表；
- 数据和元数据配置；
- 加工过程；
- 制图；
- 约束条件；
- 支持注册信息的构念。

此外，用术语“数据结构定义”取代了术语“关键字族”，是因为这两个术语是用于不同地方的同义词。本部分使用术语“数据结构定义”，见图3。

数据集	元数据集	数据和元数据供应	报告和分发						
数据结构定义	元数据结构定义	概念方案	类别方案	代码表	层级代码方案	转换和表式	结构映射	过程	结构定义
标识、项目方案、组件结构、关联									SDMX基础包

图3 SDMX 信息模型包结构

另外，针对基于场景的注册表的额外包见注册表接口规范。对于这些信息在图4中显示，其中包括：

- 订阅和通知；
- 注册；
- 发现。

注：注册功能所需的不仅限于这三个包，注册功能也需要使用信息模型中的其他包。

数据集	元数据集	数据和元数据供应	订阅和通知	注册	发现	报告和分发			
数据结构定义	元数据结构定义	概念方案	类别方案	代码表	层级代码方案	转换和表式	结构映射	过程	结构定义
标识、项目方案、组件结构、关联									SDMX基础包

图4 SDMX 信息模型（包含注册的包结构）

4 执行者和用例

4.1 执行者和用例

为了开发数据模型，有必要了解该功能，以支持从需求定义得到结果。这些定义在用例模型里。用例模型由执行者和用例组成，见下面的定义。

执行者

执行者定义一套连贯的角色，系统的用户与系统交互时可承担这些角色。执行者实例可以是个体或者外部系统。

用例

用例定义一套实例，这里每个实例是系统运行的一个序列动作，产生一个可观察的结果值给一个特定的执行者。

模型的总体目的是支持数据和元数据的报告，分发和汇总统计数据和相关元数据的字段交换。为了实现这一点，该模型需要支持这一进程的三个基本方面：

- 结构定义和供应定义的维护；
- 数据、元数据的发布（报告）和利用（使用）；
- 访问数据、元数据，以及结构定义和供应定义。

本部分包含前两个方面，关于注册表逻辑模型的标准是关于最后一个方面的。

4.2 用例图

4.2.1 维护结构和供应定义

4.2.1.1 用例

维护数据和元数据结构与供应定义的用例见图5。

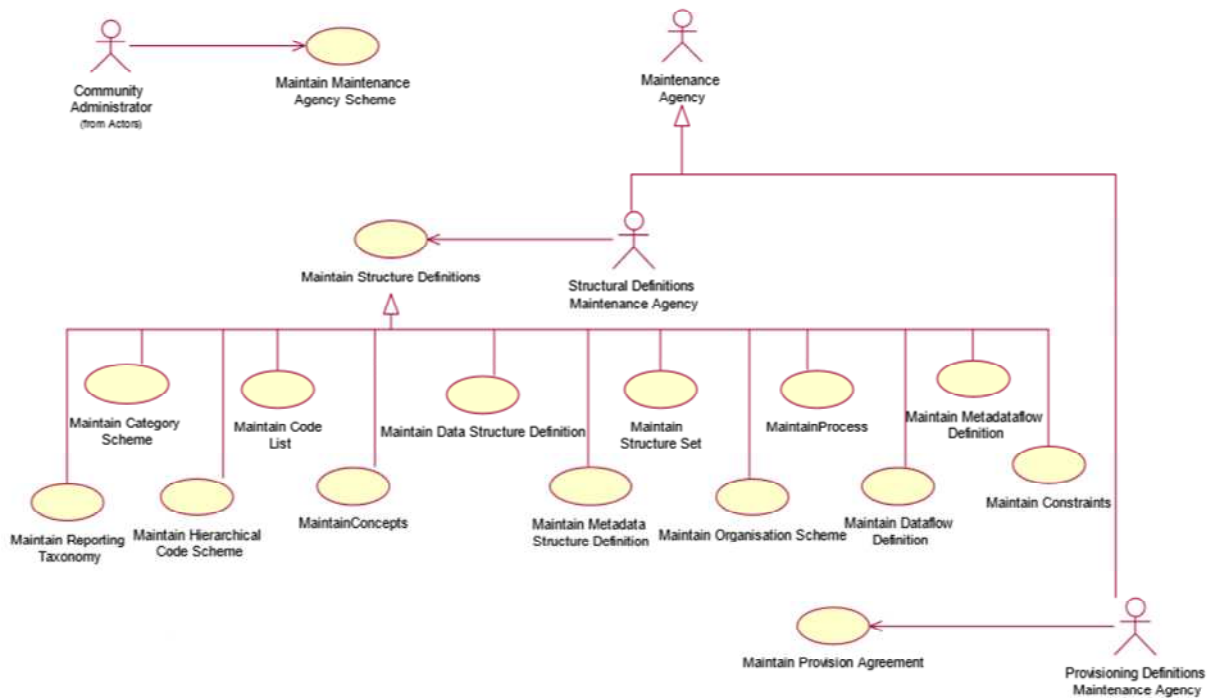


图5 维护数据和元数据结构与供应定义的用例

4.2.1.2 图解

为了应用程序发布和使用数据和元数据，必须定义结构和数据及元数据允许的内容并使之对于应用程序是可用的，就像支持实际进程的发布和使用的定义一样。这是维护机构的责任。

所有被维护的产物由维护机构维护。为了方便，维护机构执行者分为两个执行者的角色：

- 维护结构的定义；
- 维护供应的定义。


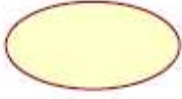


这两个功能可由同一人执行，或者至少由同一个维护组织执行，由于定义的目的不同，因此使用角色已经进行了区分：结构定义用来定义数据及元数据在被报告和分发时的格式以及被许可的内容，同时供应定义支持报告和分发的进程（报告内容，对象以及时间）。






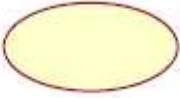


在一个基于场景的组织里面，至少结构定义可以被共享，重要的是维护机构的方案由一个尽责的组织（这里叫做Community Administrator（组织管理员））维护，因此维护代理的ID必须是唯一的。

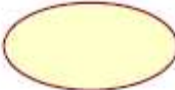






4.2.1.3 定义

针对结构和供应定义维护的执行者和用例见表2。

表2 针对结构和供应定义维护的执行者和用例表

执行者	用例	说明
 组织管理员 Community Administrator		负责管理整个组织的通用结构定义的组织。
	 维护维护机构方案 Maintain Maintenance Agency Scheme	维护机构方案的建立和维护。
 维护机构 Maintenance Agency		负责维护结构化的内容，如代码表，概念方案，数据结构定义，元数据结构定义，数据和元数据配置内容例如数据供应方和数据流定义。 子角色是： 结构定义 维护机构供应定义 维护机构
 结构定义维护机构		负责维护结构定义。

<p>Structural Definitions Maintenance Agency</p>		
	 <p>维护结构定义 Maintain Structure Definitions</p>	<p>结构定义的维护。对被维护的每个结构产物，本用例有子类用例。</p>
	 <p>维护代码表 Maintain Code List</p>  <p>维护概念方案 Maintain Concept Scheme</p>  <p>维护类别方案 Maintain Category Scheme</p>  <p>维护数据结构定义 Maintain Key Family (Data Structure Definition)</p>  <p>维护元数据结构定义 Maintain Metadata Structure Definition</p>  <p>维护立方体结构 Maintain Cube Structure</p> 	<p>数据结构定义（关键字族）的建立和维护，元数据定义和立方体结构，及他们使用的支持工具，如代码表和概念方案。</p>

	<p>维护层级代码方案 Maintain Hierarchical Code Scheme</p>  <p>维护报告分类 Maintain Reporting Taxonomy</p>	
 <p>供应定义维护机构 Provisioning Definitions Maintenance Agency</p>		<p>负责维护数据和元数据定义。</p>
	 <p>维护供应定义 Maintain Provisioning Definitions</p>	<p>供应定义的维护。对于被维护的每个结构产物，本用例有子类用例。</p>
	 <p>维护数据供应者 Maintain Data Provider</p>  <p>维护数据流定义 Maintain Dataflow Definition</p>  <p>维护元数据流定义 Maintain Metadataflow Definition</p>  <p>维护立方体定义 Maintain Cube Definition</p>	<p>工具的建立和维护，该类工具支持数据和元数据供应的定义，如数据供应方列表，数据流定义，立方体定义，及将数据供应方和数据流及元数据流连接的供应协议。</p>



4.2.2 发布和使用数据及元数据

4.2.2.1 用例

数据和元数据的发布及使用的执行者和用例见图6。

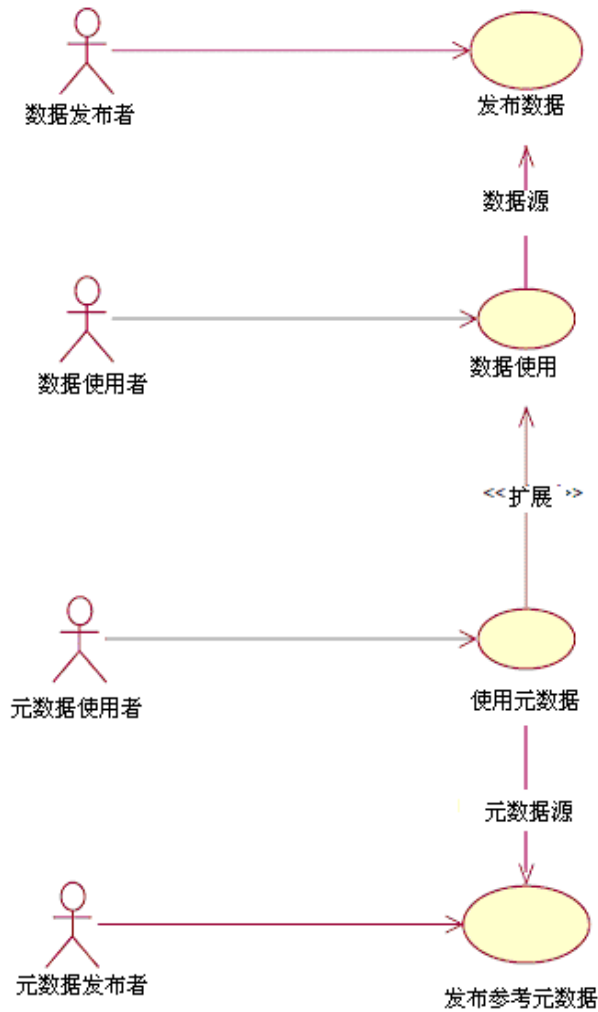


图6 数据和元数据的发布及使用的执行者和用例

注：数据和元数据的发布和使用遵照结构定义（其定义格式和允许的内容）和提供定义（其为使用者定义使数据和元数据可使用的过程）的说明

4.2.2.2 图解

注意，在本图中“发布”数据和元数据被视为和“报告”数据和元数据一样。在某些情况下，使这些数据可用的操作满足两种功能。发布汇总数据和为了使数据发布者做这些及为了使用应用程序来处理数据和元数据，必须知道其结构。此外，使用应用程序可能为了向数据访问者介绍而需要访问（引用）




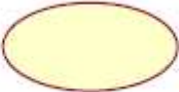

元数据，以便数据更好理解。因为有这些数据，提供的元数据也必须按照维护的结构进行格式化。数据访问者和元数据访问者直到数据和元数据发布才能使用它，因此有一个“数据源”或“元数据源”放在“使用”和“发布”用例之间。


在任何数据和元数据发布和使用的情况下，发布和使用程序都必须访问维护的供应定义。这些定义可能像谁在什么时间提供什么数据和元数据给谁一样简单，或者，由于特定的供应方提供的数据和元数据及资料来源（在数据共享的环境下，数据和元数据被从数据源“拉”出来）有约束而更复杂些。

4.2.2.3 定义

定义见表3。

表3 定义

执行者	用例	说明
 数据发布者 Data Publisher		负责按照规定的数据结构定义和有关的供应定义发布数据。
	 发布数据 Publish Data	发布一个数据集。这可能是一个物理数据集或者可能在访问数据源时使数据可用，例如一个能处理查询的数据库。
 数据使用者 Data Consumer		数据的用户。可能是一个人（消费者）通过使用界面访问，或者可能是一个应用程序，如统计生产系统。
	 使用数据 Uses Data	使用按照结构定义进行格式化的数据和按照供应定义使之可用的数据。数据通常连接到放在不同位置且被独立地发布和维护的元数据。
 元数据发布者 Metadata Publisher		负责按照规定的元数据结构定义和有关的供应定义发布参考元数据。

	 发布参考元数据 Publish Reference Metadata	发布一个参考元数据集。这可能是一个物理元数据集或者可能在访问元数据源时使元数据可用，例如一个能处理查询的元数据存储库。
 元数据使用者 Metadata Consumer		元数据的用户。可能是一个人（消费者）通过使用界面访问，或者可能是一个应用程序，如统计生产系统或分发系统。
	 使用元数据 Uses Metadata	使用按照结构定义进行格式化的元数据和通过供应定义使元数据可用的元数据。

5 SDMX 基础包

5.1 引言

SDMX基础包中的构件由支持模型中许多其它结构的基本构建块组成。因为这个原因，包里的许多类是抽象的（即仅派生的子类有实例）。

建立SDMX基础包动机如下：

- 这是标识模型中的基础原型的“最佳实践”；
- 对的常见结构和“模式”进行标识，可使得更容易理解；
- 对模式进行标识，便于重复使用。

在这章视图里的每一个类图都归类于不同的透视图的SDMX基础包。这里有具体模式的详细视图，另外，概述还给出了类之间的继承和类之间的关系。

5.2 标识、版本和维护

5.2.1 类图

SDMX标识、维护 and 版本见图7。

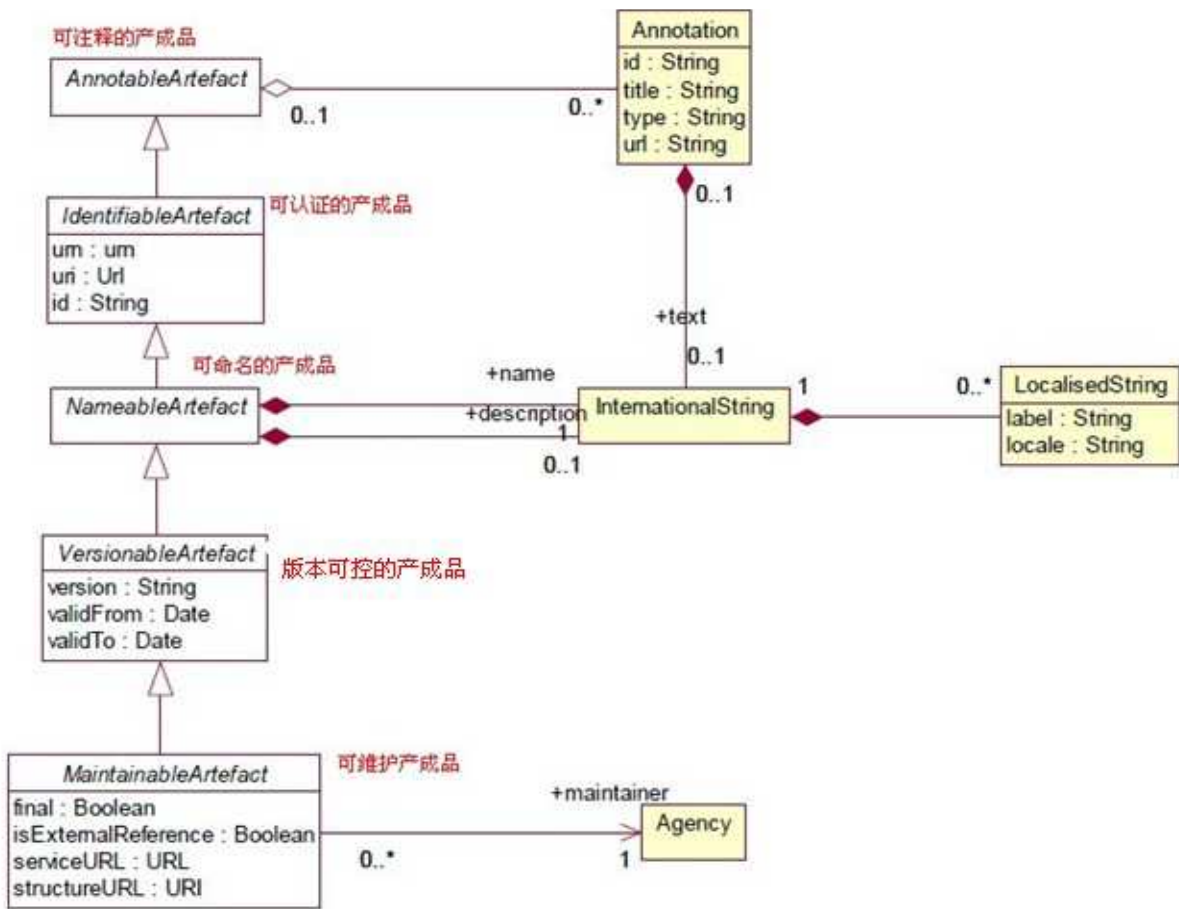


图7 SDMX 标识、维护和版本

5.2.2 图解

5.2.2.1 叙述

这一组类形成了SDMX对象管理方面的核心。它们提供了派生类可重新使用的特征如标识、版本等等。所有从抽象类AnnotableArtefact派生的类可能有注解（或注）：可为所有SDMX-ML元素增加注解。注解用于描述SDMX结构的附加信息。信息形式可能是一个URL参考文件和/或多种语言的文本（由InternationalString的关联表示）。

IdentifiableArtefact是一个抽象类，其由需要fir标识的基本属性构成。基于IdentifiableArtefact的具体类全部继承唯一标识的性能。它们亦继承带注解的性能。另外，+description和+name角色支持多语言说明和所有基于IdentifiableArtefact的对象的命名。InternationalString支持在多语言地区（地区类似于语言，但却包含地理上的差异，如加拿大法语、美国英语等）的说明的表达式。LocalisedString支持在一个地区的说明的表达式。

VersionableArtefact是一个继承自IdentifiableArtefact的抽象类且给它所有的派生类增加版本性能。

MaintainableArtefact通过到MaintenanceAgency的关联给它的派生类另外增加被维护的性能。其可能规定工具是草案还是具有final属性的终稿。

从AnnotableArtefact到MaintainableArtefact的继承链允许SDMX类继承从简单的注解到身份标识到版本和维护等它们需要的特征。

5.2.2.2 定义

定义见表4。

表4 定义

类	特征	描述
AnnotableArtefact	基本的继承子类是: IdentifiableArtefact	从它派生出的类的对象可以有附加的注释。
Annotation		一个对象的附加的描述性信息。
	Id	注释的标识符。同一个被注释的对象有多个注释时,可以区别一个注释和其它注释。
	Title	用于标识注释的标题。
	Type	指定注释如何被处理。
	url	外部描述性文本的链接。
	+text	国际化字符串,提供注释的多语言文本信息。
IdentifiableArtefact	父类是AnnotableArtefact 基本的继承子类是: NameableArtefact	可以识别所有派生类。同时对所有派生类进行注释,因为它是Annotable Artefact 的子类。
	Id	对象的唯一标识符。
	Uri	通用源标识符,可以分解或不分解。
	Urn	通用源名称——用于注册表:所有需注册的对象都有一个 urn。
NameableArtefact	父类是IdentifiableArtefact 基本的继承子类是: VersionableArtefact	除识别和注释外,还对所有派生类提供名称和描述。
	+description	通过 International String 类,提供多语言的描述。
	+name	通过 International String 类,提供多语言的名称。
InternationalString		International String 集合了各地字符串,并支持各地文本。
LocalisedString		Localised String 支持一个地区的文本描述(本地类似于语言,但包括地理变化,例如加拿大、法国、美国、英国等)。
	Label	字符串的标签。
	Locale	字符串的地理所在地,例如法国,加拿大。
VersionableArtefact	父类是 NameableArtefact 基本的继承子类是: MaintainableArtefact	对所有派生对象提供版本信息。

	Version	版本字符串, 遵从一贯规定。
	ValidForm	版本起始有效期。
	ValidTo	版本终止日期。
MaintainableArtefact	父类是VersionableArtefact	集合所有基本结构元数据内容的抽象类, 这些基本结构元数据由代理进行维护。
	Final	定义所维护内容是草稿还是最终的。
	IsExternalReference	如果设定为“true”, 表明对象内容在外部保存。
	StructureURL	SDMX-ML 文档的 URL 链接, 包括外部对象。
	ServiceURL	SDMX 网络服务 URL 链接, 外部对象可被检索到的 URL。
	+maintainer	和 Maintenance Agency 的关联, Maintenance Agency 负责维护。
Agency		参考“组织”章节。

5.3 数据类型

5.3.1 类图

基本结构的基本继承见图8, 数据类型见图9。

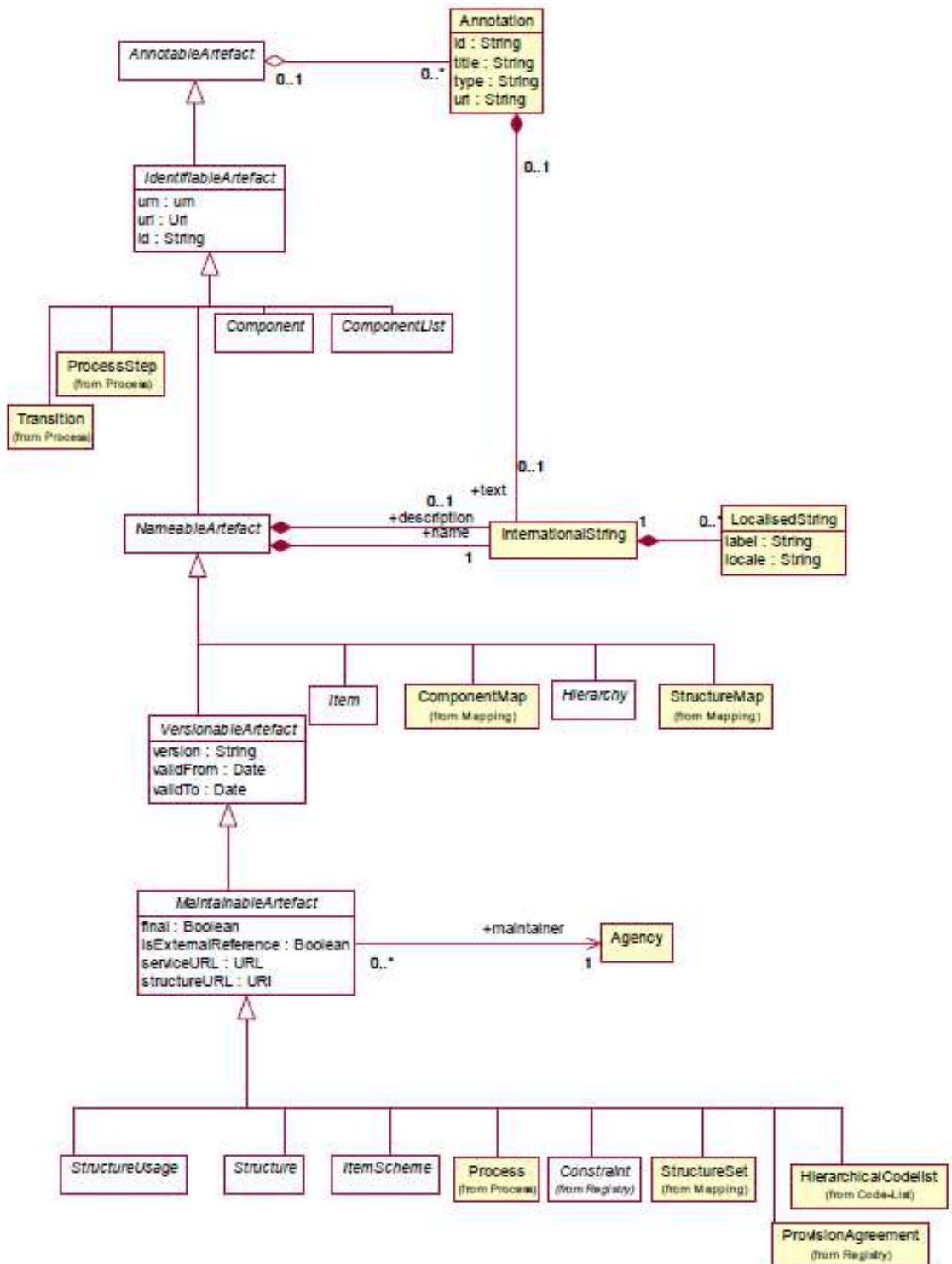


图8 基本结构的基本继承

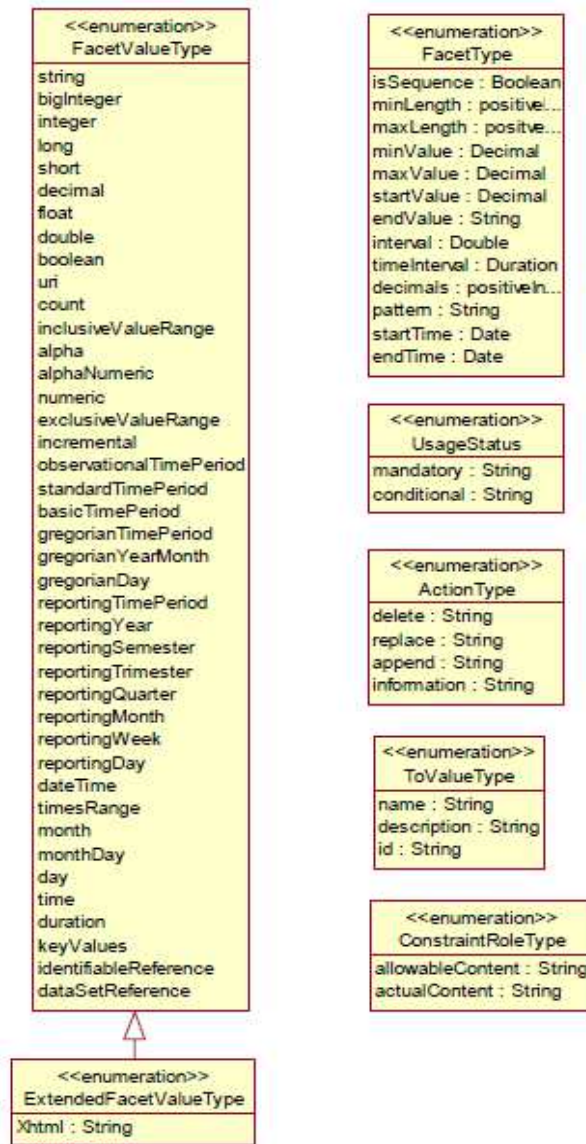


图9 数据类型

5.3.2 图解

UsageStatus 枚举用作一个属性上的数据类型，在一个类的实例中的属性值必须是 UsageStatus 的值之一（即强制、可选或有条件）。

在使用结构中的元件时（例如数据结构定义中的维度定义），面类型和面值类型的枚举用于规定一个未枚举概念内容或概念用法的有效格式。不同类型的描述可参考概念章节。

动作类型枚举用于规定某一动作的目标内容在执行时，接受系统应该进行的动作。可以枚举如下：

a) 追加

对现存的数据/元数据集或原来没有后续新增的数据或文档（属性值），数据或元数据可进行增量性的更新。如果所提供的任何数据或元数据已存在，则不会替代现存的数据或元数据。这个同SDMX技术标准1.0版本中的“Update”更新值是同样含义。

b) 替换

数据/元数据被替换，或许也包括附加的数据/元数据被添加的情况。

- c) 删除
删除数据/元数据

- d) 信息
数据和元数据都是信息。

IdentifiableObjectType 枚举用于规定对象的类型，该对象的分类是 IdentifiableArtefact 的子类，即直接由 NameableArtefact、VersionableArtefact 或 MaintainableArtefact 直接生成。

ToValueType 数据类型包括支持数据转换的属性，在 StructureMap 中有详细定义。

ConstraintRoleType 数据类型包括的属性内容有明确约束的目的（allowableContent, actualContent）。

5.4 项目方案模式

5.4.1 背景

项目方案是一个基本的结构模式，其允许创建列表方案，例如用于简单的分类系统。

ItemScheme 项目内容是以分组、代码表、概念列表、报告分类标准和组织形式等。

5.4.2 类图

Item Scheme 模式见图10。

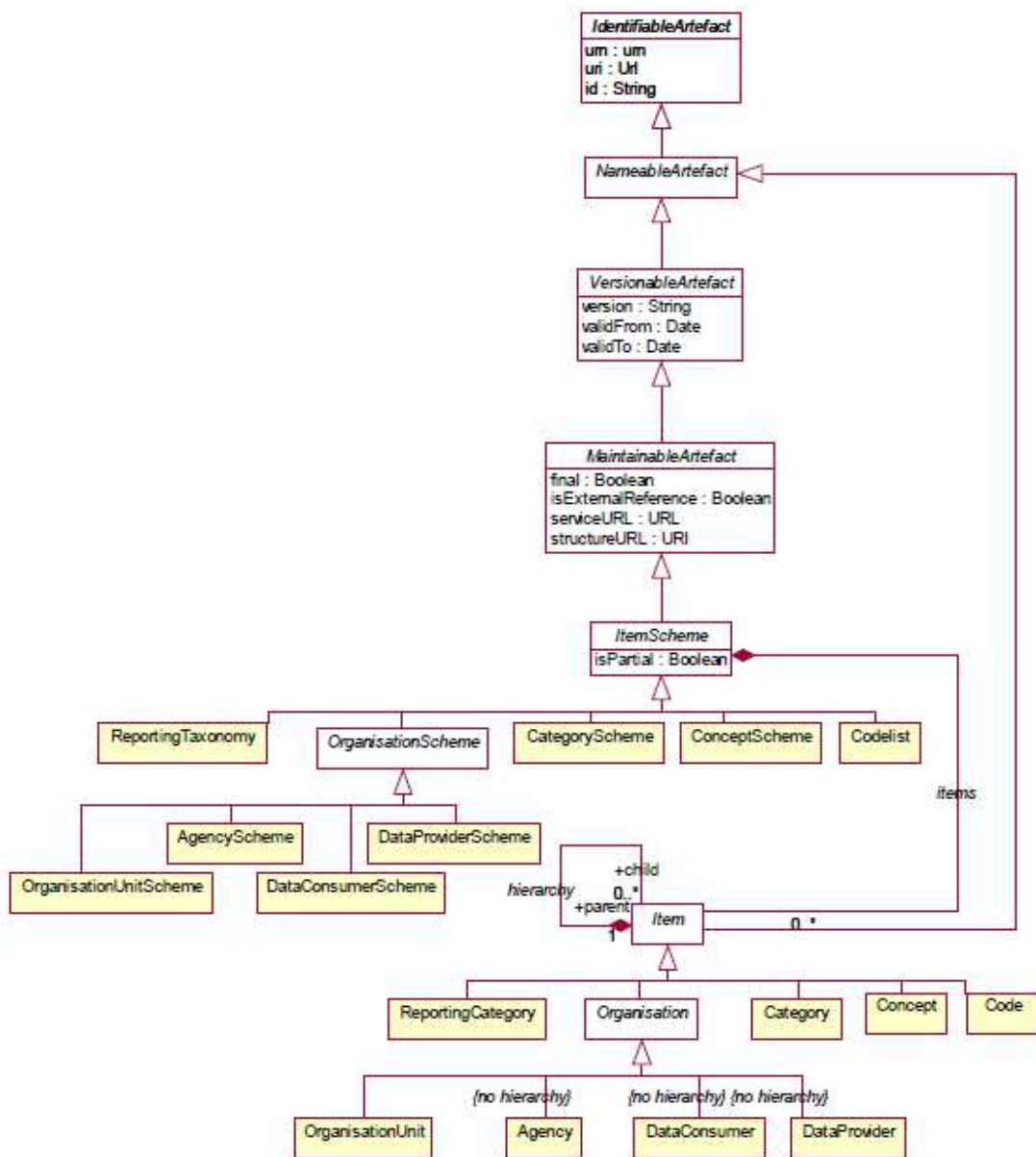


图10 Item Scheme 模式

5.4.3 图解

5.4.3.1 叙述

ItemScheme是一个抽象类，其定义了一套Item（这个类也是抽象的）。它的主要目的是定义一个机制，这个机制可以用来建立能将SDMX信息模型其它部分进行分类的分类系统。ItemScheme派生自MaintainableArtefact，赋予其进行注解、有身份标识、建立版本和与MaintenanceAgency相关联的能力。具体类的例子是CategoryScheme和关联的Category。

在交换环境下，一个ItemScheme项目允许在维护的项目列中控制一组子项目集。如果这样的项目列是由一组分散的子项目集合而成，那通过设置isPartial属性为“true”来表明项目是子项目集。

对于一个“局部的”ItemScheme在局部形势下，是不能独立被维护的。也就是说他不能包含那些在整体的ItemScheme中不存在的Items，同时任何一个Item（例如名称、描述）不能偏离在整体的ItemScheme种的内容。而且，isPartial属性是“true”的ItemScheme的ID同整体的ItemScheme（维

护的代理, id, 版本)的ID是相同的。这一点很重要, 因为这个ID是其他结构需参考的(例如一个Codelist编码列表引用自DSD) 同时这个ID会一直保持一致, 不管这个分散的 *ItemScheme* 是整体的或是局部的。

局部的 *ItemScheme* 主要支持 *ItemScheme* 子集的交换和发布, 同时不需要对众多的包含相同 *Items* 的 *ItemScheme* 进行维护。举例说明, 当一个编码列表Codelist用于数据结构定义 *DataStructureDefinition* 时, 有时只有列表中的部分子编码是相关的。在这种情况下, 运用约束机制(本文稍后会解释)可以构建局部的编码列表。

Item 继承自 *NameableArtefact*, 赋予它被注解、有标识和建立版本的能力, 因此具有ID、uri 及urn属性, 名字和说明, 其形式为 *InternationalString*。不同于模式 *ItemScheme*, *Item* 自己不是一个 *MaintainableArtefact*, 因此不能拥有独立的Agency (也就是, 它隐性地拥有与 *ItemScheme* 同样的机构)。

Item 能够被分级, 因此, *Item* 可以拥有子 *Item*。分级关联的约束是一个子 *Item* 只能拥有一个父 *Item*。

5.4.3.2 定义

定义见表5。

表5 定义

分类	特征	描述
<i>ItemScheme</i>	衍生自 <i>MaintainableArtefact</i> 直接子分类有: <i>CategoryScheme</i> <i>ConceptScheme</i> <i>Codelist</i> <i>ReportingTaxonomy</i> <i>OrganisationScheme</i>	基于对象共有的特征, 给出对象排列或分组的叙述性信息。
	isPartial	在被维护的 Scheme 中, Item Scheme 是否包含整体项目集的子集表示。
	items	Scheme 中的关联项目。
<i>Item</i>	衍生自 <i>NameableArtefact</i> 直接子分类有: <i>Category</i> <i>Concept</i> <i>Code</i> <i>ReportingCategory</i> <i>Organisation</i>	是在项目方案中的项目目录。或许是分类或本体层、编码列表中编码等的一个节点。 在组织架构概念层面的节点是不分层级的。
	hierarchy	此特征允许一个项目可随意的有一个或多个子项目。

5.5 结构模式

5.5.1 背景

结构是一个基本的模式，其允许用于统计数据中复杂的表格结构说明（如关键字族、立方体、元数据结构定义）。结构是一套有序列表。支持这种表格结构的模式已经开发出来了，因此，这些结构间的通用性能被一般的软件和一般的语法结构所支持。

5.5.2 类图

Structure模式见图11，结构模式的内在表示见图12。

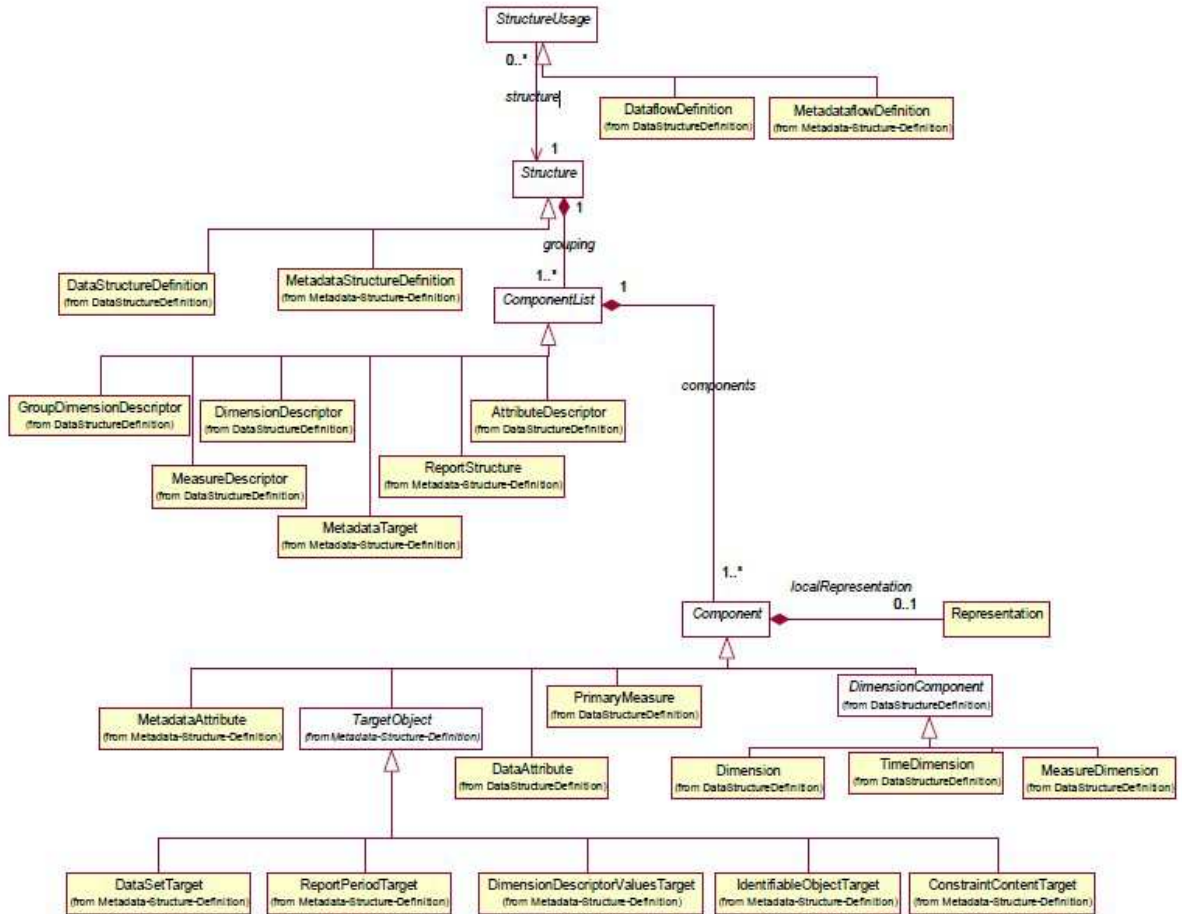


图11 Structure 模式

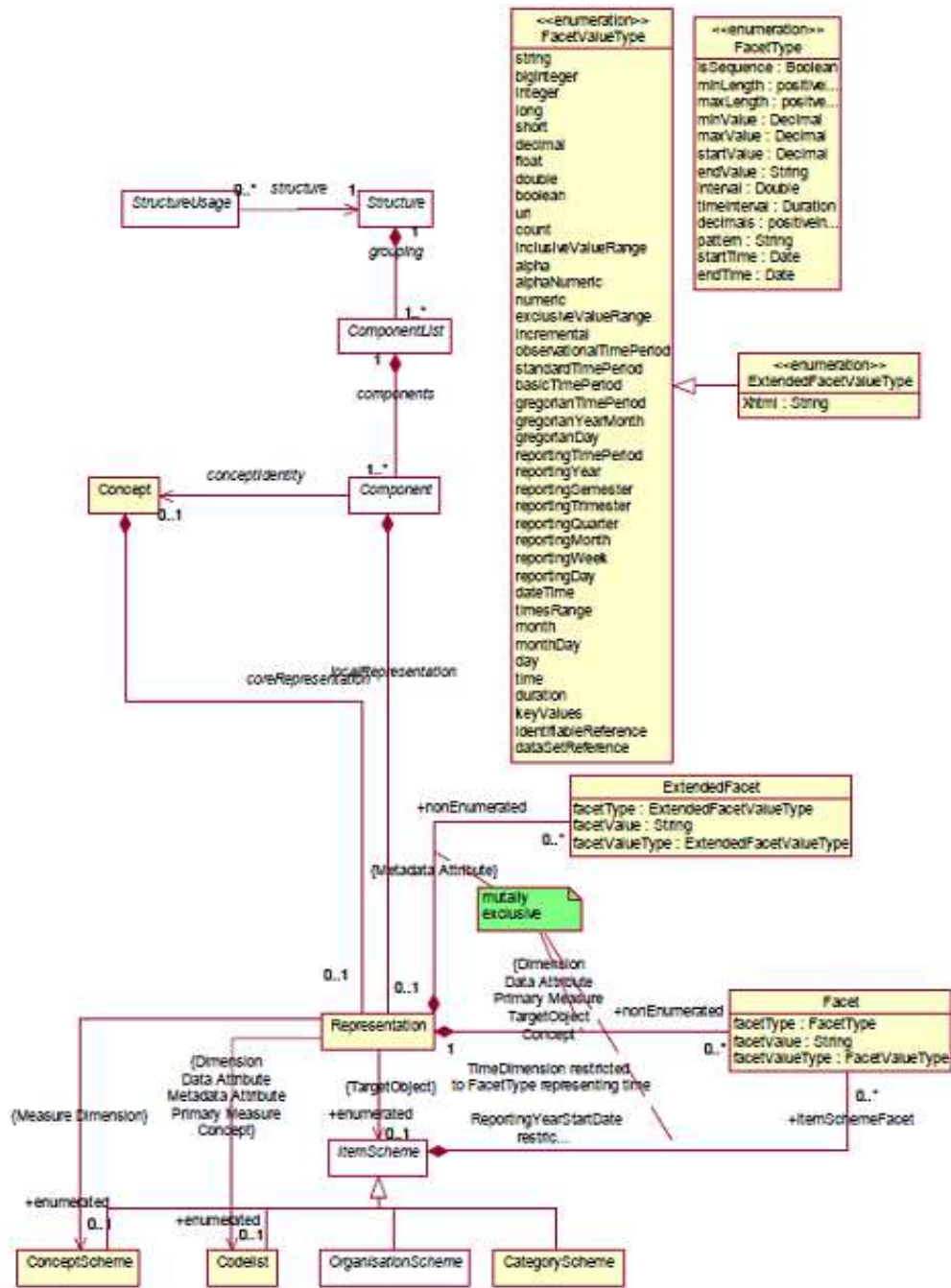


图12 结构模式的内在表示

5.5.3 图解

5.5.3.1 叙述

*Structure*是一个抽象类，它包含了一套一个或多个*ComponentList*（这个类也是抽象的）。具体的结构举例是*DataStructureDefinition*数据结构定义。

*ComponentList*是包含一个或多个*Component*的列表。*ComponentList*有几个具体描述类基于它：例如，*DimensionDescriptor* 维度描述，*GroupDimensionDescriptor* 组维度描述，*MeasureDescriptor* 估量描述，和 *AttributeDescriptor of the*

DataStructureDefinition and MetadataTarget数据结构定义和元数据指标的属性描述，以及ReportStructure of the MetaDataStructureDefinition元数据结构定义的报表结构。

Component组件包含在一个ComponentList组件列表中，其类型取决于列表的具体分类，有如下几种分类：

DimensionDescriptor维度描述：Dimension维，Measure Dimension量度维，Time Dimension时间维

GroupDimensionDescriptor组维度描述：Dimension，Measure Dimension，Time Dimension

MeasureDescriptor: PrimaryMeasure

AttributeDescriptor: Data Attribute 数据属性

MetadataTarget: TargetObject and its sub classes目标对象及其子类

ReportStructure: MetadataAttribute元数据属性

每个Component从在ConceptScheme中的一个Concept获得它的语义（同时可能也包括它的表象）。与Concept关联的conceptIdentity可作为它的代表。

Component也可能有一个localRepresentation。它允许存在一个具体的级，例如Dimension，由此规定它的代表属性，这种代表属性局部包含在Structure中（对于Dimension，它是DataStructureDefinition），因此重置任意的由Concept规定的coreRepresentation。

Representation可枚举或不可枚举。一个可枚举表示的有效内容可在ItemScheme（可以是ConceptScheme, Codelist, OrganisationScheme, CategoryScheme, 中的一个）中规定。不可枚举表象的有效内容可由一个或多个Facet规定（例如可规定最大最小值）。对于MetadataAttribute由多个Extended Facet中的一个获得，允许XHTML的附加表示。

对于具体组件有效的表示的种类在模型中用关联约束来表达，即：

- MeasureDimension必须是可枚举的，并用一个ConceptScheme
- Dimension（不包括MeasureDimension），DataAttribute, PrimaryMeasure, MetadataAttribute可以是可枚举，并如果是可枚举，需用一个Codelist
- TargetObject可以是可枚举，并如果是可枚举，需用任意一个ItemScheme(Codelist, ConceptScheme, OrganisationScheme, CategoryScheme, ReportingTaxonomy)
- Dimension（不包括MeasureDimension），DataAttribute, PrimaryMeasure, TargetObject可以是不可枚举的，如果不可枚举，需使用多个Facet中的一个，注释：适用于TimeDimension的FacetValueType仅限于表示时间。
- MetadataAttribute可以是不可枚举的，如果不可枚举，使用一个或多个ExtendedFacet
- Structure可以被一个或多个StructureUsage使用，以具体类举例，DataflowDefinition（StructureUsage的子集）可以用特定的DataStructureDefinition（Structure的子集），类似构件提供MetadataflowDefinition（与MetadataStructureDefinition关联）。

5.5.3.2 定义

定义见表6。

表6 定义

分类	特征	描述
StructureUsage	衍生自 MaintainableArtefact 子集是：	其组件由一个结构描述的产成品。 具体子集的举例如Dataflow Definition（数据流定义），与

	DataflowDefinition MetadataflowDefinition	给出的一个构件关联, 在此是指 Data Structure Definition (数据 结构定义)。
	structure	一个 Structure (构件) 的联合, 说明产成品的构件。
<i>Structure</i>	衍生自: <i>MaintainableArtefact</i> 子集: DataStructure Definition MetadataStructure Definition	负责的板状结构的抽象说明清单。 具体例子如统计概念, 代码表和在 数据或元数据结构定义中的组织, 由中间机构定义, 通常用于与合作 伙伴的统计信息交换。
	grouping	一个或多个组件清单的复合联合。
<i>ComponentList</i>	衍生自: <i>IdentifiableArtefact</i> 子集: DimensionDescriptor GroupDimension Descriptor MeasureDescriptor AttributeDescriptor MetadataTarget ReportStructure	组件清单的一个抽象定义。具体例 子如 Dimension Descriptor (维度描述符), 定义了在一个 Data Structure Definition (数据 结构定义) 中的 Dimensions 清 单。
	components	一个或多个构成清单的集合联合。
<i>Component</i>	衍生自: <i>IdentifiableArtefact</i> 子集是 PrimaryMeasure DataAttribute DimensionComponent TargetObject MetadataAttribute	组件是一个抽象的父级, 用于定义 定性的和定量的数据和元数据项, 属于 Component List (组件清单) 和由此形成的 Structure (结构)。组件由其 子级所精确化。
	conceptIdentity	在 Concept Scheme (概念图表) 中与 Concept 联合, 概念图表确 定和定义组件的语义。
	localRepresentation	组件的表示与组件使用 (ConceptUsage) 概念的核心表 示不一致时, 用于关联组件表示。
<i>Representation</i>		对于 Component 组件或 Concept 概念的允许值或格式。

	+enumerated	与可枚举清单的关联,清单包括当组件记录在数据或元数据集中的可允许的内容。对任何组件均允许的可枚举清单的类型在关联的约束中给出。(例如, Identifier Component 标示符组件可有任意的 Item Scheme 项目方案子级,然而, Measure Dimension 测量维度必须有一个 Concept Scheme 概念列表)。
	+nonEnumerated	与 Facets 的集合的关联, Facets 定义了组件内容的允许格式,用于数据或元数据集。
Facet		当组件用于数据或元数据集时定义它的内容格式。
	facetType	受 FacetType 可枚举约束的特殊内容类型。
	facetValueType	在数据或元数据中记录的 Component 组件的值格式。受 FacetValueType 可枚举约束。
	+itemSchemeFacet	定义在 Component 组件中使用的 Item Scheme 的标示符格式。典型的是定义标示符的特性数量(长度)。
ExtendedFacet		与 Facet 面向有相同功能,但允许附加的 XHTML 表示。对于与 Metadata Attribute 元数据特征使用时具有约束性。

5.5.3.3 表示构建

SDMX FacetValueTypes 的大多数与在 XML Schema 中的是相一致的。并且与目前多数的实现平台相同。

表 7 也是 SDMX 数据类型的列举,没有直接的对应,通常因为它们复合的表示或更广的数据类型的限制。这些内容在关于标准的第六章有详细阐述。

Representation 由 Facets 组成,每一个 Facet 传递的是与值域的定义相联系的特征信息。通常 Facets 的集合需要传递所需的语义。例如,一个序列有至少两个 Facets 定义:一个定义起始值,一个定义间隔。见表 8。

表7 类型属性表

SDMX Facet Value Type	XML Schema Data Type	.NET Framework Type	Java Data Type
String	xsd:string	System.String	java.lang.String
Big Integer	xsd:integer	System.Decimal	java.math.BigInteger
Integer	xsd:int	System.Int32	int
Long	xsd:long	System.Int64	long
Short	xsd:short	System.Int16	short
Decimal	xsd:decimal	System.Decimal	java.math.BigDecimal
Float	xsd:float	System.Single	float
Double	xsd:double	System.Double	double
Boolean	xsd:boolean	System.Boolean	boolean
URI	xsd:anyURI	System.Uri	Java.net.URI or java.lang.String
DateTime	xsd:dateTime	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
Time	xsd:time	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
GregorianYear	xsd:gYear	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
GregorianMonth	xsd:gYearMonth	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
GregorianDay	xsd:date	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
Day, MonthDay, Month	xsd:g*	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
Duration	xsd:duration	System.TimeSpan	javax.xml.datatype.Duration

表8 面向类型解释对照表

面向类型	解释
isSequence	表示不论值是否有序，必须与区间间隔、起始值、结束值 facet 或时间间隔、起始时间、结束时间等 facets 共同使用。如果此属性值是真，起始的值或时间和数值或时间间隔必须有。如果结束值没有给出，则序列无限可持续。
interval	指定了在一个序列中所允许的间隔（增量的）。为了使用此特征，isSequence 属性值必须为真。
startValue	与 isSequence 和 interval facets 相一致（这两个属性必须设置才能使用该属性）。用于数值型的序列，描述序列的起始点。对于表达数值型序列，值是强制的。
endValue	与 isSequence 和 interval facets 相一致（这两个属性必须设置才能使用该属性）。用于数值型的序列，描述序列的结束点。
timeInterval	表示在时间序列中所允许的持续。isSequence 的值必须为真。
startTime	与 isSequence 和 timeInterval facets 相一致（这两个属性必须设置才能使用该属性）。用于时间序列，描述序列的起始时间。对于时间序列，值是强制的。
endTime	与 isSequence 和 timeInterval facets 相一致（这两个属性必须设置才能使用该属性）。用于时间序列，描述序列的结束时间。
minLength	指定特性值的最小长度。
maxLength	指定特性值的最大长度。

minValue	用于包含和特定的值区间，描述区间的下界。如果用于闭区间，有效值必须大于或等于给出值。如果数据类型的包含关系没有给定（例如用于整数型的数据类型），在值默认为是包含在内。
maxValue	用于包含和特定的值区间，描述区间的上界。如果用于闭区间，有效值必须小于或等于给出值。如果数据类型的包含关系没有给定（例如用于整数型的数据类型），在值默认为是包含在内。
decimals	描述小数点后(十进制分隔符)特征允许的位数。
pattern	包括任何在实现语法中允许的正规表达式（例如 W3C XML Schema）。

6 具体项目方案

6.1 引言

该结构基于特征把对象安排到层或列表里，并把它当成继承自 *ItemScheme* 的组来维护。这些具体的类是：

- CodeList;
- ConceptScheme;
- CategoryScheme;
- AgencyScheme;
- DataProviderScheme;
- DataConsumerScheme;
- OrganisationUnitScheme.

这些全部衍生于抽象分类 OrganisationScheme Reporting Taxonomy。

6.2 继承视图

图 13-19 的每一幅图里都展示了继承和关系视图。

6.3 代码表

6.3.1 类图

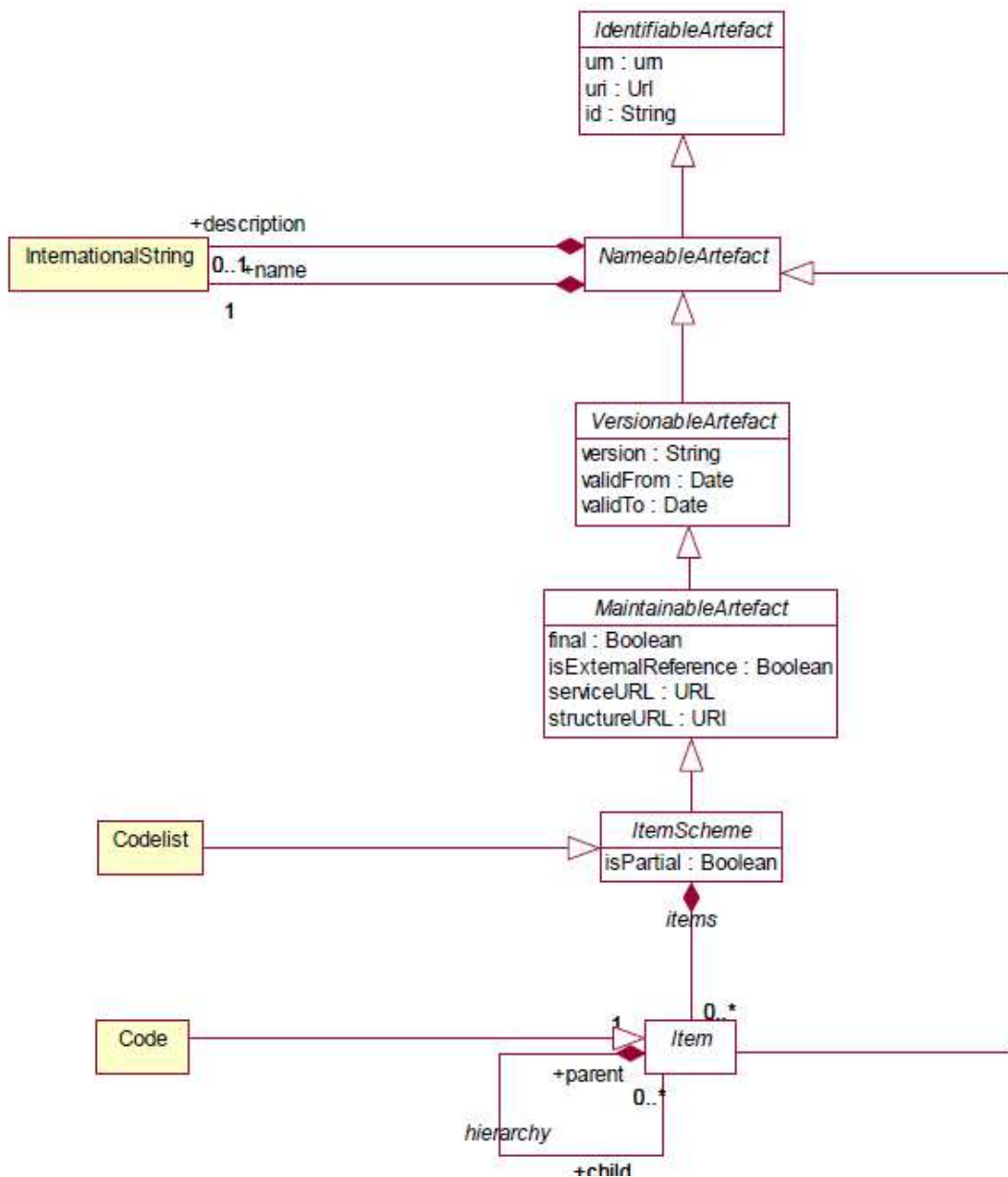


图13 代码表 (Code List) 的类图

6.3.2 图解

6.3.2.1 叙述

CodeList继承自ItemScheme，因此有以下属性：

- id;
- uri;
- urn;
- version;
- validFrom;
- validTo;
- isExternalReference;


```

---serviceURL;
---structureURL;
---final;
---isPartial.

```

Code 衍生自 *Item* 并有如下属性:

```

---id;
---uri;
---urn.

```

它们也关联到 *InternationalString* 以支持多语言名字, 有一个可选的多语言说明, 并且有一个到 *Annotation* 的关联以支持注解 (没有展示)。

通过继承, *CodeList* 由一个或多个 *Code* 组成, 而 *Code* 本身在 *hierarchy* 关联中可以有一个或多个子 *Code*。注意, 在此关联中一个子 *Code* 有且仅有一个父 *Code*。稍后说明的更复杂的 *CodeSet* 和 *HierarchicalCodeScheme* 允许有多个父类和多个层次。*HierarchicalCodeScheme* 中的 *Code* 引用自 *HierarchicalCodeScheme*, 但是可能要求从 *Code* 连接到一个 *HierarchicalCodeScheme* 中的 *Hierarchy* (这样一个连接支持代码映射—见第9章), 且支持经由 *hierarchyView* 关联。

6.3.2.2 定义

定义见表9。

表9 定义

类	特征	说明
<i>CodeList</i>	继承自 <i>ItemScheme</i>	若干统计概念(代码概念)从中取值的一个列表。
<i>Code</i>	继承自 <i>Item</i>	字母、数字或符号的语言独立集, 表示一个概念, 概念的含义在自然语言中描述。
	<i>/hierarchy</i>	关联父代码和子代码。

6.4 概念方案

6.4.1 继承类图

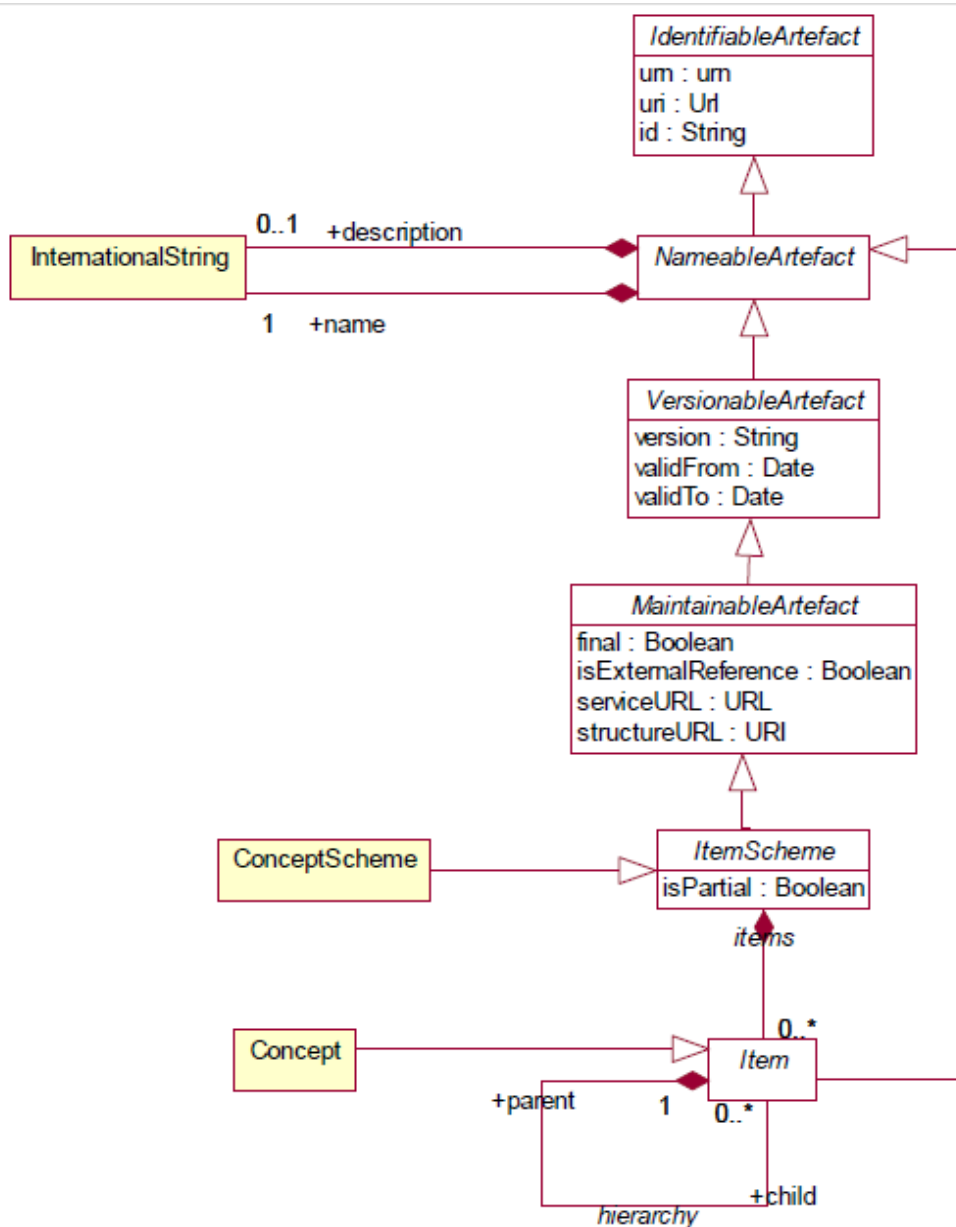


图14 概念方案（Concept Scheme）的类图

6.4.2 图解

ConceptScheme继承自ItemScheme，因此有以下属性：

- id;
- uri;
- urn;
- version;
- validFrom;
- validTo;
- isExternalReference;
- registryURL;
- structureURL;

—repositoryURL;
 —final;
 —isPartial.

Concept 衍生自 Item 并有如下属性:

—id;
 —uri;
 —urn.

虽然衍生自 NameableArtefact, 它们也关联到 InternationalString 以支持多语言名字, 有一个可选的多语言说明, 并且有一个到 Annotation 的关联以支持注解 (没有展示)。

虽然由 ItemScheme 衍生而来, 但 ConceptScheme 包含一个或多个 Concepts, 并且 Concept 本身可有一个或多个子 Concepts 在 (所继承的) hierarchy 层级联合中。注意一个子 Concept 只能在关联中有一个父 Concept。

局部的 ConceptScheme (isPartial 为真) 与一个 ConceptScheme 相同, 包含 Concept 和关联的名称和描述, 正像在普通的 ConceptScheme 概念图表中一样。然而, 它的内容是所有 ConceptScheme 的子集合。

6.4.3 关系类图

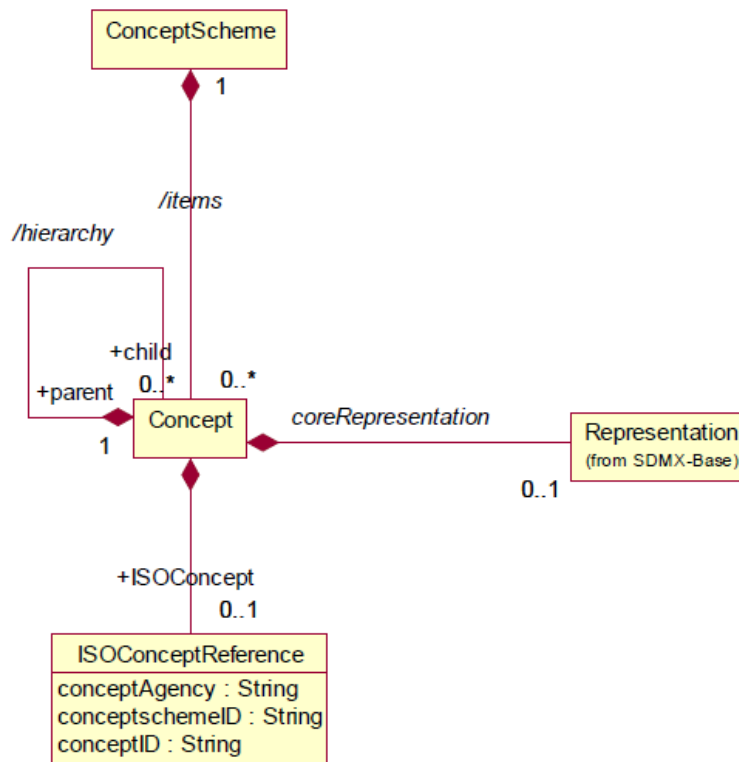


图15 概念方案 (Concept Scheme) 的关系类图

6.4.4 图解

6.4.4.1 叙述

ConceptScheme 可以拥有一个或更多的 Concept。一个 Concept 可以有 0 个或多个的子 Concept, 因此它支持一个 Concept 层次。注意, 在此关联中一个子 Concept 有且仅有一个父

Concept。层次的目的是把具有语义关系的概念联系起来：例如，一个Reporting_Country和Vis_a_Vis_Country两者都有一个作为父概念的Country，或者一个CONTACT可能有一个PRIMARY_CONTACT作为子概念。定义报告结构不是该方案的目的：这些报告结构定义在MetadataStructureDefinition里。

Concept能被定义成符合指定的Type，就像字符串，数字串等等，这是它的一个coreType且它可能也有一个指定的Representation—coreRepresentation即：当用在一个结构像KeyFamily或MetadataStructureDefinition时，除非Type或Representation的说明在有关的结构定义中被替换，coreType和coreRepresentation是Concept的格式和值域的说明。在一个分级的ConceptScheme中，除非在子Concept等级上被替换，Type和Representation继承自父Concept。

注意 ConceptScheme 在 DataStructureDefinition 中用于 MeasureDimension 的 Representation (见5.3.2)。每一个在ConceptScheme中的Concept是一个具体估量，每个都可赋予一个 coreRepresentation。因此每个估量的观测值的有效格式当出现在在 MeasureDimension的数据集中时，都在Concept中规定。允许每个估量有不同的格式。

Representation的详细文字描述在基于SDMX的章节中。

Concept或许与在ISO/IEC 11179标准中描述的一个概念有关联。

ISOConceptReference识别这一概念和包含的概念图表。

6.4.4.2 定义

定义见表10。

表10 定义

分类	特性	描述
ConceptScheme	衍生自 <i>ItemScheme</i>	基于特性将概念排列或分割为组的描述性信息，基本特性为目标对象所共有的。
Concept	衍生自 <i>Item</i>	由唯一的特性组合创建的知识单元。
	/hierarchy	关联父级概念和子级概念。
	coreRepresentation	关联 Representation。
	+ISOConcept	关联一个 ISO 概念参照。
ISOConceptReference		一个 ISO 概念定义的身份。
	conceptAgency	包含概念的概念图表的维护代理。
	conceptSchemeID	概念图表的标示符。
	conceptID	概念的标示符。

6.5 类别方案

6.5.1 背景

这个程序包定义了类图中支持分类的定义和分类间关系的结构。类似概念图表程序包。类图的一个例子是对数据分类的图表—有时是作为主题领域的图表或数据分类列表。重要的是，正如稍后会看到的，在设计图表中的单个注释可与任何在Categorisation中的IdentifiableArtefacts集关联。

6.5.2 类图

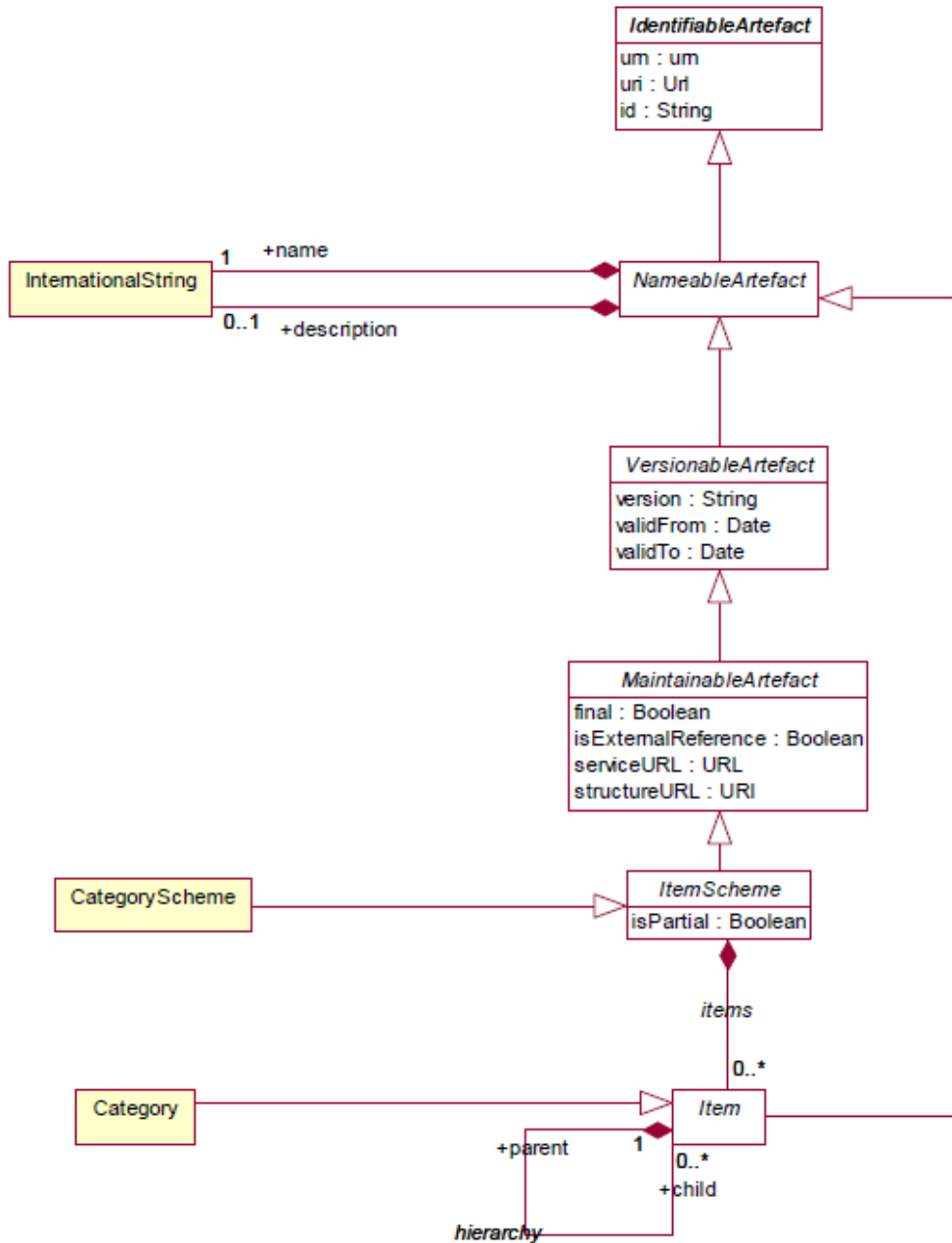


图16 类别方案 (Category Scheme) 的类图

6.5.3 图解

类别被建成一个分层的 *ItemScheme*。CategoryScheme 继承自 *ItemScheme*，有以下的属性：

- id;
- uri;
- urn;
- version;
- validFrom;
- validTo;
- isExternalReference;

—structureURL;
 —serviceURL;
 —final;
 —isPartial.

Category 由 Item 衍生而来并具有如下特性:

—Id;
 —uri;
 —urn.

两个也都关联到InternationalString以支持多语言名字, 有一个可选的多语言说明, 并且有一个到Annotation的关联以支持注解(没有展示)。

CategoryScheme可以有一个或多个Category, 一个Category可以有0个或多个子Category, 以支持Category的层次。注意, 在此关联中一个子Category有且仅有一个父Category。

局部的CategoryScheme (isPartial为真) 与一个CategoryScheme相同, 包含Category和关联的名称和描述, 正像在普通的CategoryScheme概念图表中一样。然而, 它的内容是所有CategoryScheme的子集合。

6.5.4 类图

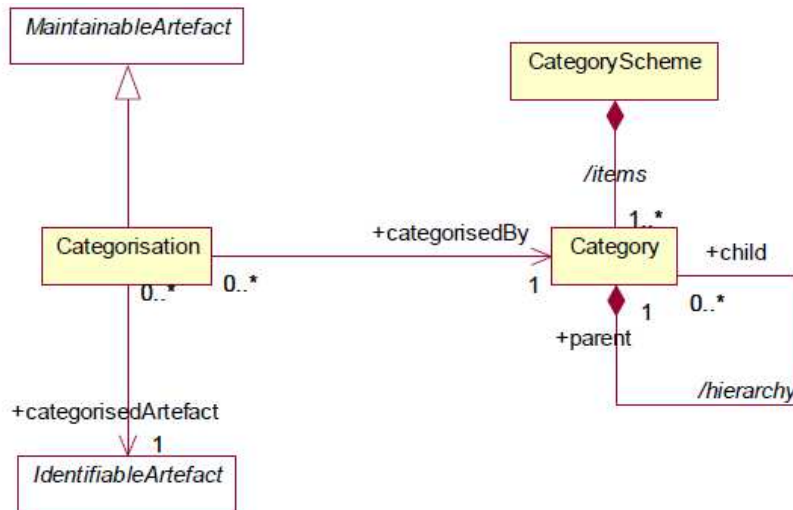


图17 对象类型方案 (Object Type Scheme) 的类图

CategoryScheme可有一个或多个Categories。Category是Identifiable可认证的并有认证信息。Category可以有零个或多个子级的Categories, 因此支持多层的Categories。任何IdentifiableArtefact可认证的产成品可被Category分类+categorisedBy。通过Categorisation实现。每个Categorisation都与某个有一个Categorisation的IdentifiableArtefact关联。多重的Categorisations可用于建立同样Category下被+categorisedBy的IdentifiableArtefacts集。注意从Category到Categorisation是无法联通的(即没有嵌入的参考)。由实现透视图有必要使得Categorisation对Category或IdentifiableArtefact的版本控制无影响。

定义见表11。

表11 定义

分类	特性	描述
CategoryScheme	衍生自	基于特性将分类排列或分割为组

	<i>ItemScheme</i>	的描述性信息,基本特性为目标对象所共有的。
	/items	联合分类。
Category	衍生自 <i>Item</i>	在任何级下的项目,包括等级分类、标准表格分类、章节、小节、分部、细分、分组、子分组、类、子级等。
	/hierarchy	关联父分类和子分类。
Categorisation	衍生自 <i>MaintainableArtefact</i>	将 <i>IdentifiableArtefact</i> 与一个 <i>Category</i> 关联。
	+categorisedArtefact	关联 <i>IdentifiableArtefact</i> 。
	+categorisedBy	关联 <i>Category</i> 。

6.6 组织方案

6.6.1 类

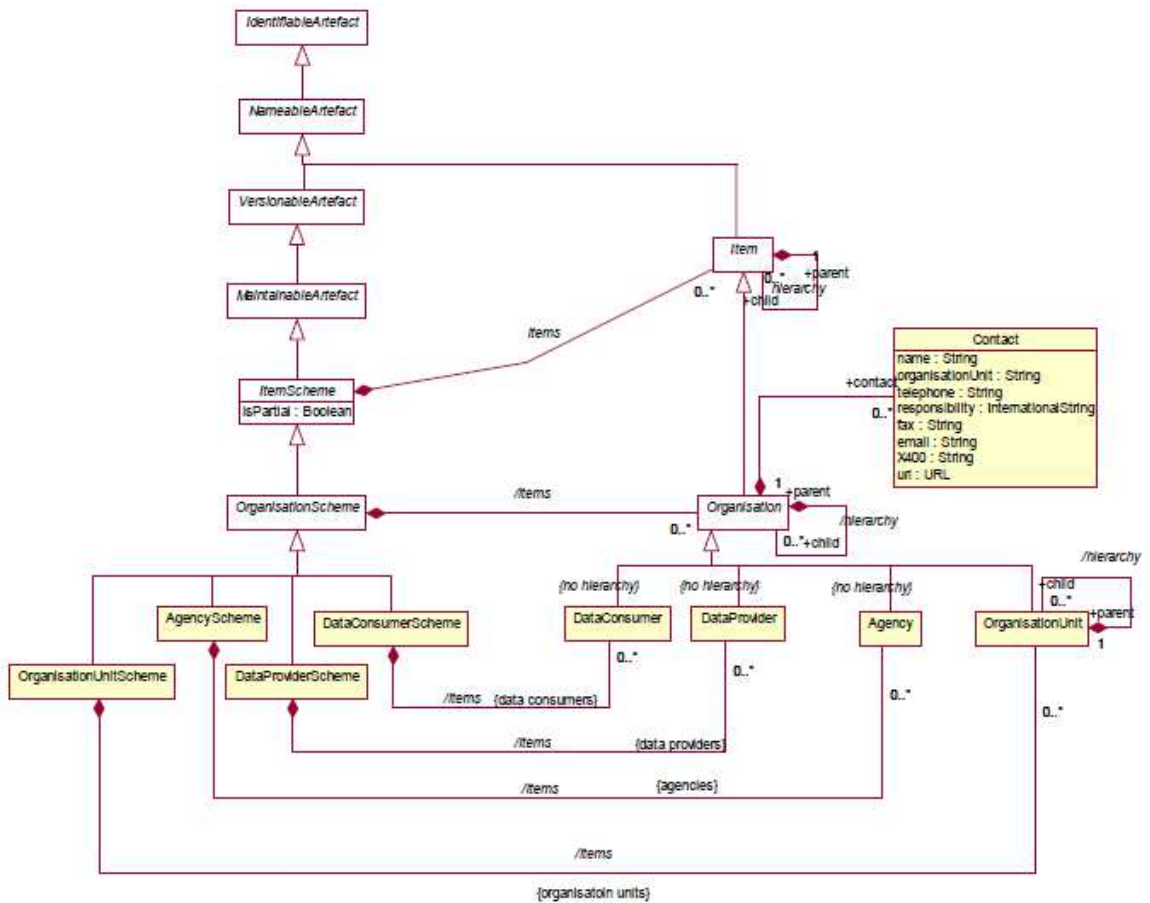


图18 组织类图

6.6.2 图解

6.6.2.1 描述

*OrganisationScheme*是抽象的，它包含的*Organisation*也是抽象的。

*Organisation*可有子*Organisation*。

*OrganisationScheme*可以是四种类型之一：

- a) *AgencyScheme* -包含局限于代理的平面列表的*Agency*（即没有层级）。注意SDMX的维护代理(Maintenance) *Agency*系统可分层，在单独的文件“Technical Notes”中将有详细的解释；
- b) *DataProviderScheme*-包含局限于代理的平面列表的*DataProvider*（即没有层级）；
- c) *DataConsumerScheme*-包含局限于代理的平面列表的*DataConsumer*（即没有层级）；
- d) *OrganisationUnitScheme*-包含由*Organisation*继承的 /hierarchy 关联性的 *OrganisationUnit*。

参考元数据可借助元数据附属机制附属于 *Organisation*。该机制在本文的Reference Metadata参考元数据章节有解释。这意味着模型除了有限的Contact信息外，不特别说明可附属于 *DataProvider*、*DataConsumer*、*OrganisationUnit* 或 *Agency*的具体参考元数据。

局部的*OrganisationScheme* (*isPartial*为真) 与一个*OrganisationScheme*相同，包含 *Organisation*和关联的名称和描述，正像在普通的*OrganisationScheme*概念图表中一样。然而，它的内容是所有*OrganisationScheme*的子集合。

6.6.2.2 定义

定义见表12。

表12 定义

类	特性	描述
<i>OrganisationScheme</i>	抽象类 衍生自 <i>ItemScheme</i> 子级有： <i>AgencyScheme</i> <i>DataProviderScheme</i> <i>DataConsumerScheme</i> <i>OrganisationUnitScheme</i>	<i>Organisations</i> 组织的维护合集。
	/items	在 scheme 设计图中与 <i>Organisations</i> 的关联。
<i>Organisation</i>	衍生自 <i>Item</i> 子级有： <i>Agency</i> <i>DataProvider</i> <i>DataConsumer</i> <i>OrganisationUnit</i>	组织有唯一的权威架构，在此架构中一个人或多个人基于某种目的进行动作或被指定动作。
	+contact	对 Contact 联系信息的关联。
	/hierarchy	对子 <i>Organisations</i> 的关联。
<i>Contact</i>		个人或组织(或组织的部分或组织中的人)角色的实例，对于这个角

		色在某个特定环境下可以发送或提取信息项, 实质对象和/或人。
	name	用语言表达式对 Contact 联系人的指定。
	organisationUnit	用语言表达式对组织架构的指定, Contact 联系人在此工作。
	responsibility	相对于组织角色的联系人职责, 是人的 Contact 目的。
	telephone	Contact 的电话号码。
	Fax	Contact 的传真号码。
	email	Contact 的互联网邮件地址。
	X400	Contact 的 X400 地址。
	uri	Contact 的 URL 地址。
AgencyScheme		Maintenace Agencies 维护代理的维护合集。
	/items	在设计图中与 Maintenace Agencies 的关联。
DataProviderScheme		Data Providers 数据提供者的维护合集。
	/items	在设计图中与 Data Providers 的关联。
DataConsumerScheme		Data Consumers 数据使用者的维护合集。
	/items	在设计图中与 Data Consumers 的关联。
OrganisationUnitScheme		Organisation Units 组织单位的维护合集。
	/items	在设计图中与 Organisation Units 的关联。
Agency	衍生自 Organisation	维护产成品的责任代理, 包括统计分类, 专业术语、结构元数据例如 Data and Metadata Structure Definitions 数据和元数据结构定义, Concepts 概念和 Code lists 代码表。
DataProvider	衍生自 Organisation	提供数据或参考元数据的组织。
DataConsumer	衍生自 Organisation	为进一步加工而输入使用数据的机构。
OrganisationUnit	衍生自 Organisation	组织结构中的指派。

	/hierarchy	与子 Organisation Units 组织单元的连接。
--	------------	--------------------------------

6.7 项目方案关联

6.7.1 类图

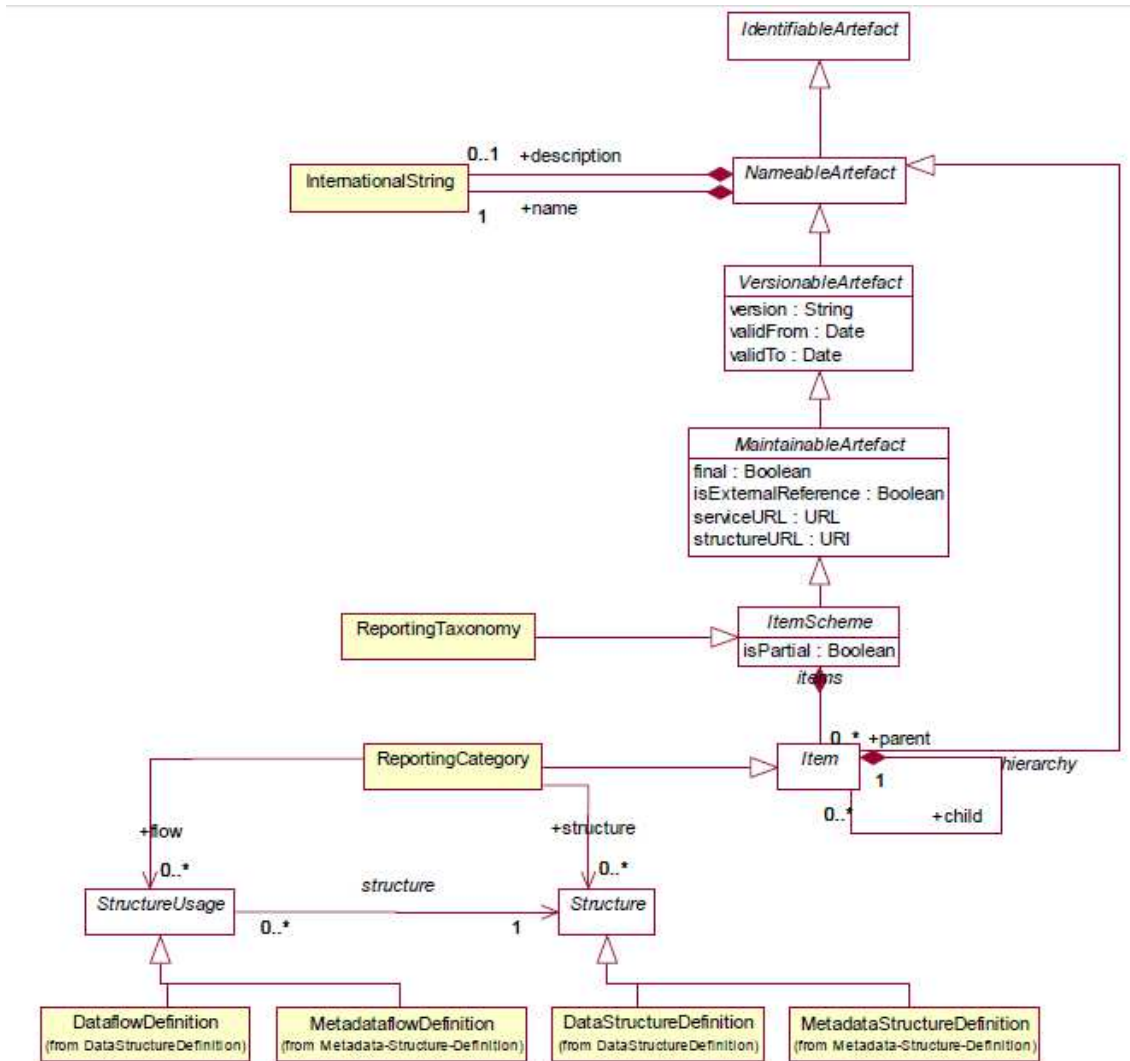


图19 项目方案关联 (Item Scheme Association) 的类图

6.7.2 图解

6.7.2.1 叙述

在某些数据报表环境里，特别是哪些主要报表中，一个报告或许包括各种异类数据，每种被一个不同的Structure描述。同样的，一个具体的发布或印刷的报告或许也包括各种异类数据。一系列相连接子报告可以由ReportingTaxonomy来定义。

ReportingTaxonomy是ItemScheme的一个特别形式。ReportingTaxonomy的每个ReportingCategory可以链接到一个或多个属于DataflowDefinition或MetadataflowDefinition之一的StructureUsage上，也可链接到一个或多个属于

DataStructureDefinition或MetadataStructureDefinition的Structure上。在特定的ReportingTaxonomy中,每个Category通过这种方式链接的将会与相同层级链接(例如在设计图中所有的Category将链接到一个DataflowDefinition)。注意ReportingCategory可有子ReportingCategory并且通过这种方法可以定义一个多层级的ReportingTaxonomy。在分类法中定义某些ReportingCategory仅仅是为了给出报告结构,这是可行的,例如:

Section 1

- a) linked to DataflowDefinition_1;
- b) linked to DataflowDefinition_2.

Section 2

- a) linked to DataflowDefinition_3;
- b) linked to DataflowDefinition_4.

在此Section 1和Section 2的节点不会链接到DataflowDefinition,但其他的会链接到一个DataflowDefinition(以及由此的DataStructureDefinition)。

局部的ReportingTaxonomy(isPartial为真)与一个ReportingTaxonomy相同,包含ReportingCategory和关联的名称和描述,正像在普通的ReportingTaxonomy概念图表中一样。然而,它的内容是所有ReportingTaxonomy的子集合。

6.7.2.2 定义

定义见表13。

表13 定义

类	特性	描述
ReportingTaxonomy	衍生自 ItemScheme	定义一个数据报表的构成结构的设计图,报表中的每个组件可被独立的Dataflow Definition 数据流定义或Metdataflow Definition 元数据流定义所描述。
	items	与 Reporting Category 报表分类的关联。
ReportingCategory	衍生自 Item	给出报表结构并将其链接到数据和元数据的组件。
	hierarchy	与子 Reporting Category 报表分类的关联。
	+flow	与数据和元数据流的关联,链接到有关提供的和相关的数据和元数据集的元数据,以及定义它们的结构。
	+structure	与 Data Structure Definition 数据结构定义和 Metadata Structure Definitions 元数据结构定义的关联,定义了描述包

		含在报表这个部分的数据和元数据 的结构元数据。
--	--	----------------------------

7 数据结构定义和数据集

7.1 介绍

DataStructureDefinition是数据的结构定义的一个类名称。有些组织将这类定义识别为“Key Family”，因此这两个名称是类似的。本说明中使用Data Structure Definition这个属于（也简称DSD）。

模型中在这个层的很多构念是从SDMX Base Layer中衍生出来的。因此有必要学习所继承的和相关的图表以便理解单个程序包的功能。在简单子模型中，这些内容可在相同图表中列出，但是为了内容清晰，在更复杂的子模型中是被删掉的。基于这些原因，下面的继承图表将所有与数据结构定义相关的分类均包含在了整个继承树中。

在这个子模型中，相比下面的继承图表，仅非常少的附加类。换句话说，SDMX Base按照关联和属性给出了这种子模型的大部分结构。本章的关联图表清晰的显示出从SDMX Base衍生的关联性。（查看附录A中用于描述图解注释）。

可衍生出具体类的实际的SDM Base架构取决于类的如下需求：

- Annotation 注释 - AnnotableArtefact;
- Identification 身份识别 - IdentifiableArtefact;
- Naming 命名 - NameableArtefact;
- Versioning 版本控制 - VersionableArtefact;
- Maintenance 维护 - MaintainableArtefact.

7.2 继承视图

7.2.1 类图

数据结构定义和数据集包中的类继承见图20。

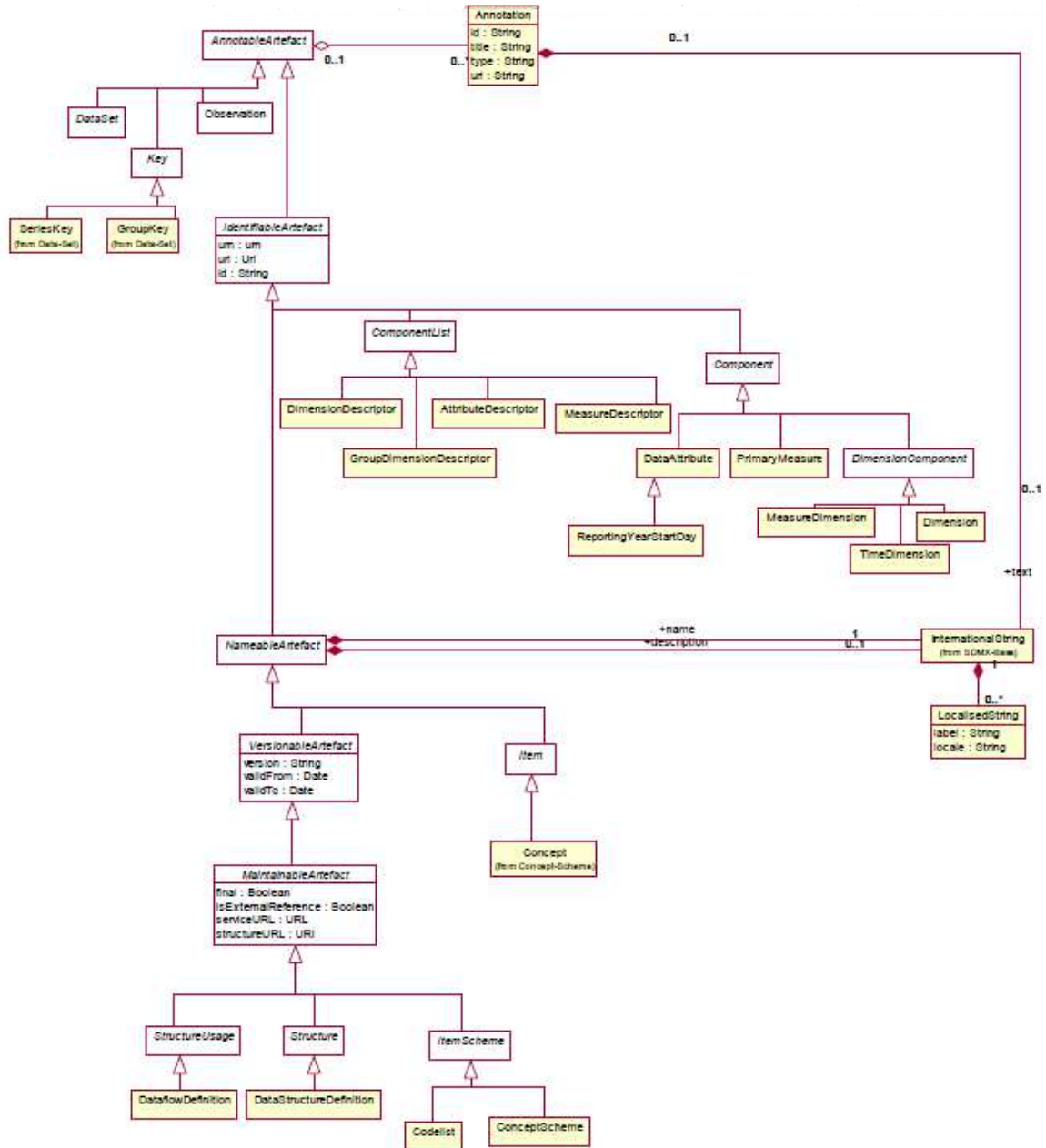


图20 数据结构定义和数据集包中的类继承

7.2.2 图表解释

在SDMX需要注释的元数据模型中的类衍生自 *AnnotableArtefact*。类有：

- *IdentifiableArtefact*;
- *DataSet* (and therefore *StructureSpecificDataSet*, *GenericDataSet*, *GenericTimeSeriesDataSet*, *StructureSpecificTimeSeriesDataSet*);
- *Key* (and therefore *SeriesKey* 和 *GroupKey*).

在SDMX需要注释和全球认证的元数据模型中的类衍生自 *IdentifiableArtefact*，它们是：

- *NameableArtefact*;

- ComponentList;
- Component.

在SDMX需要注释、全球认证、多语言命名和多语言描述的元数据模型中的类衍生自NameableArtefact, 它们是:

- VersionableArtefact;
- Item.

在SDMX需要注释、全球认证、多语言命名和多语言描述、以及版本管理的元数据模型中的类衍生自VersionableArtefact, 它们是:

- MaintainableArtefact.

代表信息的抽象类均衍生自MaintainableArtefact, 这些信息由Maintenance Agencies维护, 这些类也继承了VersionableArtefact的所有特征:

- StructureUsage;
- Structure;
- ItemScheme.

上述所有类都是抽象的。理解本章类图的关键是衍生自抽象类的具体类。

这些在元数据模型的SDMX Data Structure Definition and Dataset包中的具体类均衍生自MaintainableArtefact (相比较其他抽象类而言), 元数据模型需由Agencies代理维护, 它们是:

- DataflowDefinition;
- DataStructureDefinition.

组件结构, 即清单的列表, 直接衍生自Structure。一个Structure包括若干组件清单。具体的衍生自Structure的类是:

- DataStructureDefinition;
- 一个DataStructureDefinition 包含一个维度清单、一个层组清单和一个熟悉清单。

衍生自ComponentList 且是DataStructureDefinition的子组件的具体类是:

- DimensionDescriptor-内容是Dimension, MeasureDimension 和 Time Dimension;
- DimensionGroupDescriptor-内容是Dimension, MeasureDimension、Time Dimension 的关联;
- MeasureDescriptor-内容是 PrimaryMeasure;
- AttributeDescriptor-内容是 DataAttribute.

衍生自Component的类有:

- PrimaryMeasure;
- DimensionComponent 以及它的子级 Dimension, MeasureDimension、和 Time Dimension;
- DataAttribute.

衍生自DataAttribute的类是:

- ReportingYearStartDay.

以上可标识的具体类是为DataStructureDefinition定义元数据模型所需要的主要类。

本章其余部分的图和解释展示这些具体类是如何关联的以便支持所需要的功能。

7.3 数据结构定义——关联视图

7.3.1 类图

数据结构定义关系类图见图 21。

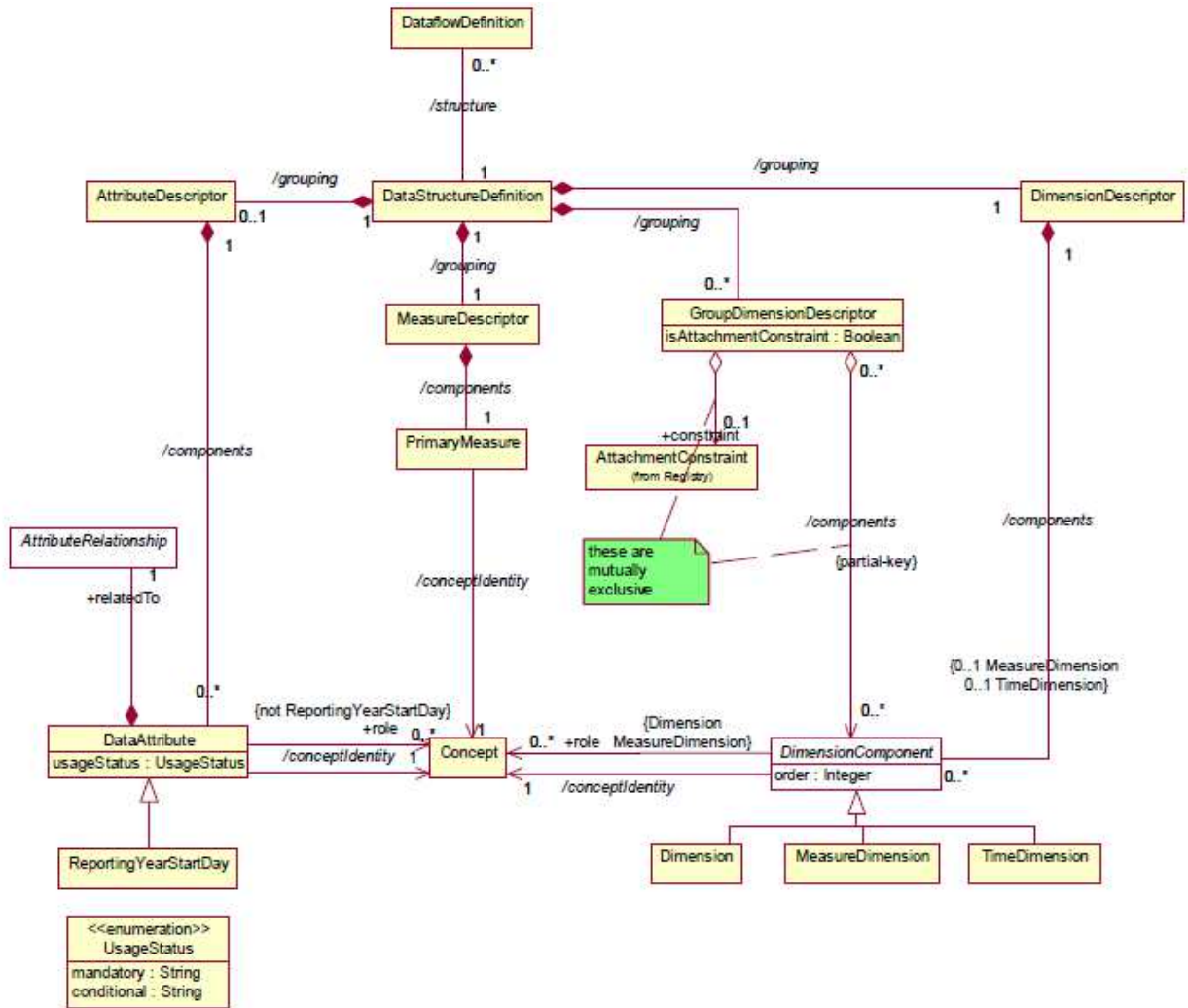


图21 数据结构定义关系类图（不包括表示）

7.3.2 图解

7.3.2.1 叙述

DataStructureDefinition定义Dimensions, MeasureDimension, TimeDimension, DataAttributes和PrimaryMeasure, 以及关联的Representation, 它包括在DataSet数据集中包含的数据的有效结构和相关属性。数据集由DataflowDefinition定义。

DataflowDefinition 或许还有附加的定义定性信息的元数据, 以及使用DataStructureDefinition的Constraints约束, 例如用在一个Dimension中的Codes子集(稍后会在文中涉及-参看“Data Constraints and Provisioning”)。每个DataflowDefinition指定一个DataflowDefinition的最大值, 用于定义任何被报到/分发的DataSets数据集的结构。

以下是维度的三个类型, 每种均与Concept有一个共同的关联:

- Dimension;
- MeasureDimension;
- TimeDimension.

注意在这里的描述中，DimensionComponent可以是任意或所有的它的子类。即Dimension, MeasureDimension, TimeDimension, 同时术语“DataAttribute”数据属性指的是DataAttribute和它的子级ReportingYearStartDate。

DimensionComponent, DataAttribute和PrimaryMeasure链接到定义其名称和语意的Concept (/conceptIdentity与Concept关联)。DataAttribute, Dimension和MeasureDimension (但不是TimeDimension)可随意的有一个与认证其在DataStructureDefinition中的角色的Concept关联的+conceptRole。因此, Concept允许的角色在ConceptScheme中被维护。角色的举例是: geography地理, entity实体, count计数, unit of measure计量单位。这些角色的运用是为了实现有意义的方式下处理数据的功能(例如给地图矢量赋一个维度值)。通常机构(例如官方统计部门)会在组织内容协调这些角色, 由此在组织中通过有意义的方式实现数据的交换和共享。

当在DataStructureDefinition中使用时, DimensionComponent、PrimaryMeasure或DataAttribute的有效值是由Representation定义得出的, 除非在DataStructureDefinition (localRepresentation)中它被重写-见图19。注意对于MeasureDimension, Representation必须是一个ConceptScheme, 并且必须总是参照MeasureDimension并由此对于与/conceptIdentity关联的Concept的Representation是不能缺省的。同时注意TimeDimension和ReportingYearStartDate受特定FacetValueTypes的约束。

总是存在一个DimensionDescriptor的集合, 可以识别所有包含所有关键信息的Dimension。同时Dimensions详细说明了Observation的关键点。

DimensionComponent可随意的按多重的GroupDimensionDescriptors (分组维度描述符)分组, 每个GroupDimensionDescriptors可识别形成部分关键点的Dimensions的组。GroupDimensionDescriptor必须被识别(GroupDimensionDescriptor.id), 同时用于DataSet的GroupKey中以便明确在DataSet中的这个group level组平面哪些DataAttributes是要记录报告的。

在DimensionDescriptor中或许会指定一个MeasureDimension的最大值。MeasureDimension的目的是正式地指定量度的含义(因为PrimaryMeasure典型的有一个通用的含义例如观测值)并使用多重量度可在StructureSpecificDataSet中被定义和公告。注意MeasureDimension参考ConceptScheme作为它的Representation(后文可见), 然而Dimension可以有可枚举(Codelist)的或不可枚举(Facet)表示。对于MeasureDimension在ConceptScheme中的Concepts组成了允许量度的清单。这样使得每一个单独量度的代表性作为Concept的coreRepresentation是清晰的, 由此对于这个MeasureDimension Concept的观测值按照PrimaryMeasure的规定重新定义其Representation。

在DimensionDescriptor中可指定一个TimeDimension的最大值。这个TimeDimension用于规定在数据集中用于明确观察时间区间的Concept。TimeDimension必须包含一个时间的有效值, 不能使用代码。

PrimaryMeasure是可观测的现象, 并且虽然可能只有唯一的PrimaryMeasure, 为保证与ComponentList/Component模式的一致性可依据MeasureDescriptor进行分组。

DataAttribute定义了所收集或分发的数据的特性, 并根据单一的AttributeDescriptor在DataStructureDefinition中做分组。DataAttribute可因命令或条件而指定, 如同在usageStatus中定义。在结构中DataAttribute或许是一个特定的角色, 同时依据与识别角色的Concept相关联的+role而给出规定。

作为+relatedTo的AttributeRelationship，定义在一个DataSet数据集中目前DataAttribute需公告的构念，指定DataAttribute。当与以下产成品相关时，DataAttribute可被指定。

- DataSet数据集 (NoSpecifiedRelationship);
- Dimension or set of Dimensions维度或维度集 (DimensionRelationship;
- 依据GroupKey而指定Set of Dimenshions维度集 (GroupRelationship-可因兼容性而保留-或DimensionRelationship的+groupKey);
- Observation观测值 (PrimaryMeasureRelationship) .

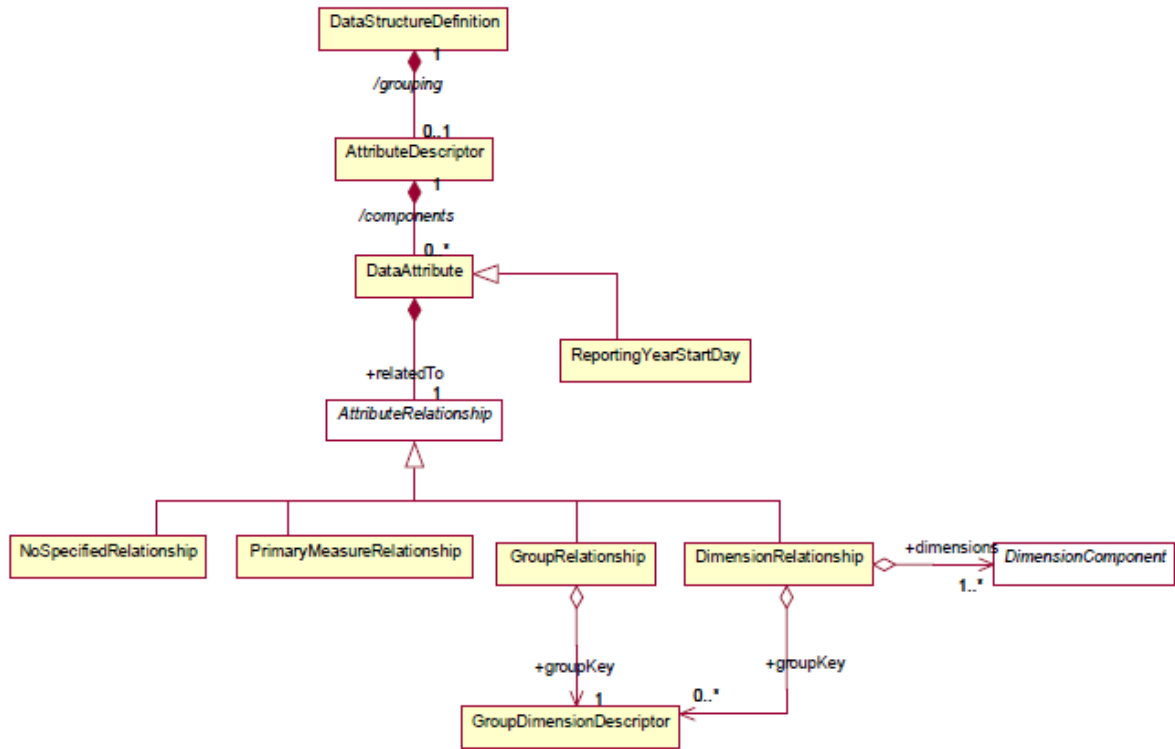


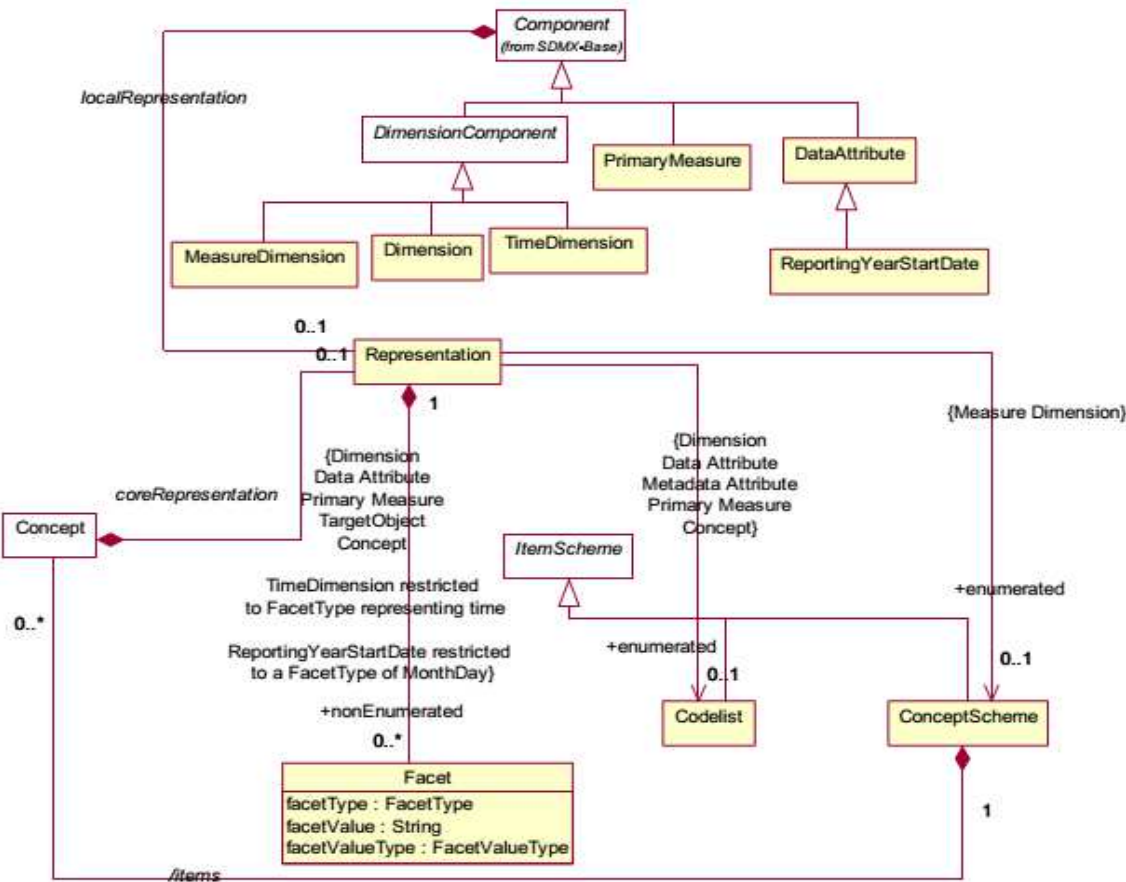
图22 数据结构定义中的属性

表14详细给出了DataAttribute可能指定的关系。这些相关性是相互排斥的，并且因此以下仅有一项是可能的。

表14 相关性

相关性	意义	在 Data Set 数据集中的 Attribute 属性被记录的位置
None	属性的值不随其他 Component 组件的值变化。	属性记录在Dataset Attribute 层面。
Dimension (1..n)	属性值将随着参考 Dimension(s) 维度值而变化。在这种情况下，属性所在的 Group(s) 组或许是随意指定的。	属性记录在Attribute属性相关的Dimension维度的最底层，另外，如果Attachment Group(s) 附属组被指定则记录在Group组所在的层面。
Group	Attribute 属性值随着 Group 组中包含的所有 Dimensions 维度值的集合而变。增加此项是为了方便列出所有 Dimensions 维度和附属的 Group 组，但使用的条件是仅当 Attribute 属性值基于所有 Group Dimension 组维度值而变	该属性记录在 Group 组所在层面。

	化。	
Primary Measure	属性值随观测值而变。	属性记录在 Observation 观测所在的层面。



<<enumeration>> FacetType	
isSequence	: Boolean
minLength	: positiveInteger
maxLength	: positiveInteger
minValue	: Decimal
maxValue	: Decimal
startValue	: Decimal
endValue	: String
interval	: Double
timeInterval	: Duration
decimals	: positiveInteger
pattern	: String
startTime	: Date
endTime	: Date

<<enumeration>> FacetValueType	
string	
bigInteger	
integer	
long	
short	
decimal	
float	
double	
boolean	
uri	
count	
inclusiveValueRange	
alpha	
alphaNumeric	
numeric	
exclusiveValueRange	
incremental	
observationalTimePeriod	
standardTimePeriod	
basicTimePeriod	
gregorianTimePeriod	
gregorianYearMonth	
gregorianDay	
reportingTimePeriod	
reportingYear	
reportingSemester	
reportingTrimester	
reportingQuarter	
reportingMonth	
reportingWeek	
reportingDay	
dateTime	
timesRange	
month	
monthDay	
day	
time	
duration	
keyValues	
identifiableReference	
dataSetReference	

图23 DSD 组件表示

DSD 组件表示见图 23。

Dimension, MeasureDimension, TimeDimension, PrimaryMeasure, 和 DataAttribute, 任意一个都能有一个指定的 Representation (使用 localRepresentation 关联)。如果在 DataStructureDefinition 中没有指定, 那么将使用指定给 Concept (coreRepresentation) 的表示。对于 MeasureDimension, 对单个度量的表述在 MeasureDimension 参照的 ConceptScheme 中为 Concept 指定。

DataStructureDefinition 可延展以形成一个派生的 DataStructureDefinition。这一点在 StructureMap 中被支持。

7.3.2.2 定义

定义见表15。

表15 定义

类	特性	描述
StructureUsage		见“SDMX Base”。
DataflowDefinition	衍生自 StructureUsage	数据流的抽象概念 (即没有数据的结构), 提供者将因不同的参照阶段提供。
	/structure	将 Dataflow Definition 数据流定义关联到 Data Structure Definition 数据结构定义。
DataStructureDefinition		元数据概念的合集, 当用于搜集或分发数据时这些概念的结构和用法。
	/grouping	与元数据概念集的关联, 这些概念在 Data Structure Definition 数据结构定义中有被认定的结构角色。
Group DimensionDescriptor	衍生自 ComponentList	元数据概念集, 定义在 Data Structure Definition 数据结构定义中源于 Dimension Descriptor 维度描述的部分关键点。
	+constraint	识别指定子级的 Attachment Constraint 附属约束, 包括 Dimension 维度、Measure 量度、或可附属 Attribute 属性的 Attribute 属性值的子级。
	/components	与包含组的 Dimension 维度和 Measure Dimension 量度维组件的关联。
DimensionDescriptor	衍生自	元数据概念的有序集, 组合统计

	ComponentList	系列分组，并且赋值，当实际组合（关键点）时，例如一个数据集，唯一识别一个指定观测值。
	/components	与 Dimension 维度、Measure Dimension 量度维度和 Time Dimension 时间维度的联合，包括 Key Descriptor 关键描述符。
AttributeDescriptor	衍生自 ComponentList	元数据概念集，定义一个 Data Structure Definition 数据结构定义的属性。
	/components	与 Data Attribute 数据属性组件的关联。
MeasureDescriptor	衍生自 ComponentList	一个元数据概念，定义 Data Structure Definition 数据结构定义的量度。
	/components	与量度组件的关联。
Dimension	衍生自 Component	用于分类统计系列的元数据概念（多数情况与其他元数据概念同时使用），统计系列举例如表示某个经济实体或一个地理参考范围的统计概念。
	/role	与 Concept 概念的关联，概念指定 Dimension 维度在 Data Structure Definition 数据结构定义中担任的角色。
	/conceptIdentity	与定义 Dimension 维度语义的元数据概念的关联。
MeasureDimension	衍生自 Dimension	统计概念，识别在关键结构中的组件，结构有一个可枚举的量度清单。此维度因其表现有枚举量度概念的 Concept Scheme 概念设计。
TimeDimension	衍生自 Dimension	元数据概念，识别“时间”关键结构中的组件。
DataAttribute	衍生自 Component 子级 ReportingYear StartDay	对象或实体的属性。
	/role	与在 Data Structure Definition 数据结构定义中指定 Data Attribute 数据属性角色的

		Concept 概念的关联。
	usageStatus	定义使用状态，受 Usage Status 使用状态数据类型约束。
	+relatedTo	与 Attribute Relationship 属性相关性的关联。
	/conceptIdentity	与定义组件语义的 Concept 概念的关联。
ReportingYearStartDay	衍生自 DataAttribute	特定化的 Data Attribute 数据属性，其值用于配合在 Time Dimension 时间维中预先定义的报告期。如果未给出值，那么默认所有基于 Time Dimension 时间维的报告期值将假设为报告年度的开始，即 1 月 1 日。
PrimaryMeasure	衍生自 Component	元数据概念，在数据集中被度量的现象。在数据集中，度量的实例通常称为观测。
	/conceptIdentity	与承载量度值的 Concept 概念的关联。
AttributeRelationship	抽象类 子级 NoSpecified Relationship PrimaryMeasure Relationship GroupRelationship Dimension Relationship	指定产成品的类型，在 Data Set 数据集中可附属 Data Attribute 数据属性。
NoSpecifiedRelationship		Data Attribute 数据属性不与任何指定架构相关。
PrimaryMeasure Relationship		Data Attribute 数据属性与 Primary Measure 基本量度结构相关。
GroupRelationship		Data Attribute 数据属性与 Group Dimension Descriptor 组维度描述符结构相关。
	+groupKey	与 Group Dimension Descriptor 组维度描述符的关联。
DimensionRelationship		Data Attribute 数据属性与 Dimensions 维度集相关。
	+dimensions	与 Data Attribute 数据属性相关的 Dimensions 维度集的关联。

	+groupKey	与指定 Data Attribute 数据属性附属的 Dimensions 维度集的 Group Dimension Descriptor 组维度描述符的关联。
--	-----------	--

关于类、属性和关联包括Representation表示的解释在SDMX Base章节中描述。

7.4 数据集——相关性视图

7.4.1 背景

数据集包含数据值的合集以及根据一个已知的DataStructureDefinition而搜集或分发的关联元数据。

7.4.2 类图

数据集类图见图24。

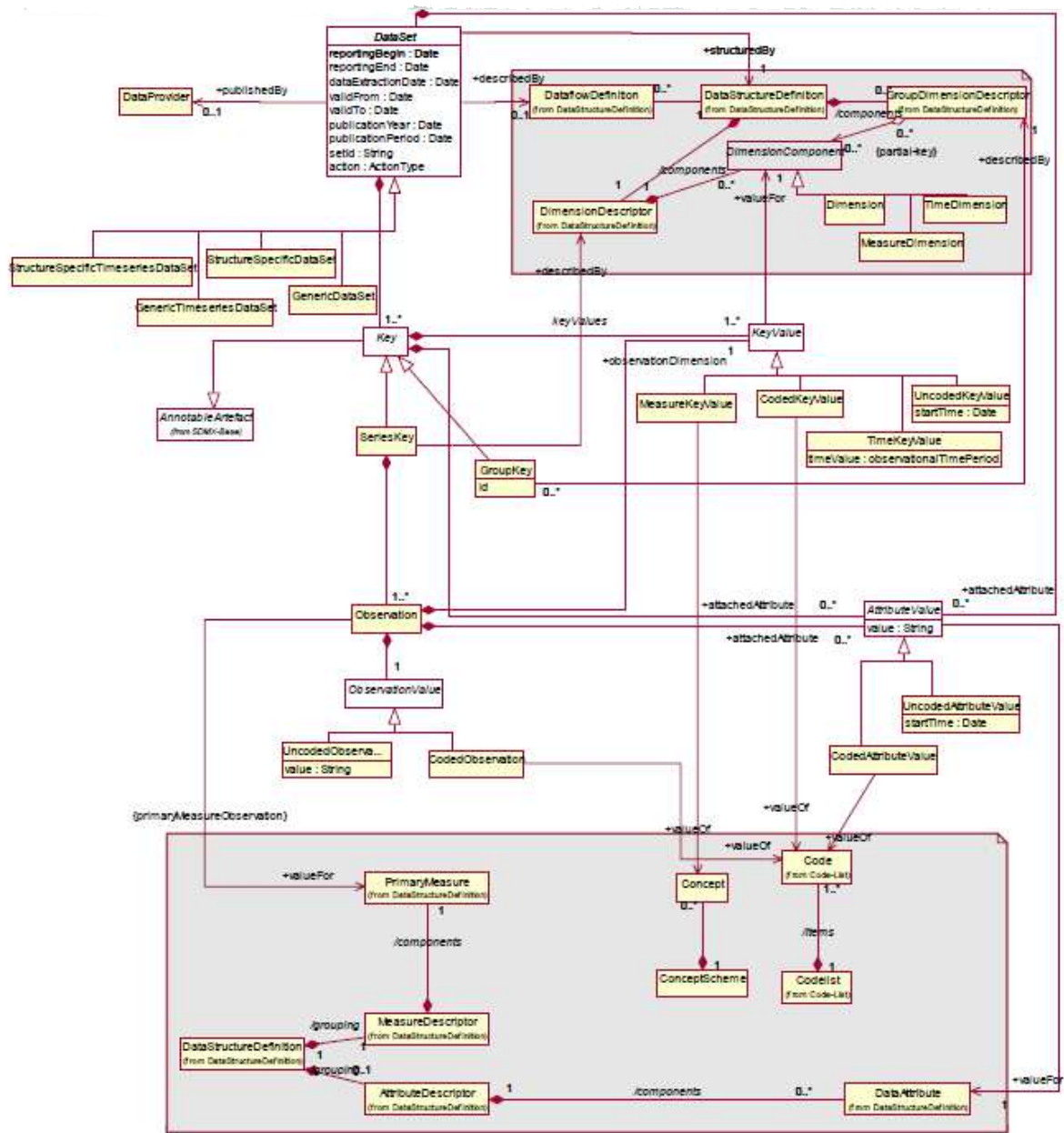


图24 数据集类图

7.4.3 图解

7.4.3.1 叙述-数据集

注意，DataSet 数据集必须符合与 DataflowDefinition 数据流定义相关联的 DataflowDefinition 数据结构定义，由此，此 DataSet 数据集是“数据的实例”。同时模型描述了 DataStructureDefinition 的子级的关联，这是为了描述链接到 DataStructureDefinition 的概念性目的。在实际的 DataSet 数据集中，为实现交换，显然必须有参照的 DataflowDefinition 和可选的 DataflowDefinition，但 DataflowDefinition 不需要与数据交换。因此，DataflowDefinition 的子级在灰色区域显示，如同当 DataSet 数据集交换时，它们不是 DataSet 数据集的一部分。然而在 DataflowDefinition 中的结构元数据可用于验证 DataSet 数据集的内容，

依据在DataflowDefinition数据结构定义中由Representation定义的KeyValue关键值的有效内容来验证。

担任DataProvider数据提供者角色的组织可对一个或更多的DataSet数据集负责。

一个数据集可格式化为一般的数据集(GenericDataSet、GenericTimeseriesDataSet)或一个DataStructureDefinition数据结构定义的特定数据集(StructureSpecificDataSet, StructureSpecificTimeseriesDataSet)。不论DataSet数据集表示的DataStructureDefinition数据结构定义如何,一般型数据集的结构都按同样方式定义。结构型的数据集根据一个指定的DataStructureDefinition数据结构定义构建。基于实现功能所选择的语法,结构型数据集需要在语法面更支持检验。

DataSet数据集是共享同样维度的Observation观测值集的合集,这由在DataStructureDefinition的DimensionDescriptor中定义的唯一组件集(Dimension, MeasureDimension, TimeDimension)来制定,同时有关联的属性值,定义附属产成品的具体特征。-DataSet, Observation, DimensionS集。数据集依据ObservationS观测报告提交的SeriesKey系列关键构建。

Observation观察值可以是度量Concept概念的变量值,概念与在DataStructureDefinition数据结构定义的MeasureDescriptor度量描述中的PrimaryMeasure基本度量相关联。当没有MeasureDimension度量维时可以指定每一个Observation观测值的精确含义。每一个Observation观测都与一个有着KeyValue(+observationDimension)关键值的ObservationValue观测值关联,关键值是为“Dimension at the Observation Level观测平面的维度”设置的。任何维度可作为“Dimension at the Observation Level观测面的维度”而指定,同时这种指定是在DataSet数据集层面做出的(即不许与整个DataSet数据集有同样的维度)。

如果“Dimension at the Observation Level观测面的维度”是MeasureDimension度量维,那么可以(但不是强制)在MeasureDimension度量维所参照作为Representation表示的ConceptScheme概念组中,用一个或多个Concept概念的明显识别记录一次Observation观测。换句话说,实际的ConceptS概念是明确的建立在Observation观测基础上的。

如果需要明确的指定DataSet数据集是时间序列,那么需要使用GenericTimeSeriesDataSet或StructureSpecificTimeSeriesDataSet中的一个同时+observationDimension观测维度的KeyValue关键值必须是一个TimeKeyValue时间关键值。在GenericDataSet和StructureSpecificDataSet中允许+observationDimension观测维度是任意的维度,包括TimeDimension时间维。

KeyValue关键值是在DataStructureDefinition数据结构定义中明确指定的MeasureDimension度量维、TimeDimension时间维或Dimension维度三者之一的值。如果是维度,关键值可(CodedKeyValue)编码或(UncodedKeyValue)未编码。如果是MeasureDimension度量维,则关键值是MeasureKeyValue度量关键值。如果是TimeDimension时间维,则是TimeKeyValue时间关键值。CodedDimensionValue编码维度值能用的实际值必须是在Codelist代码表中的Codes代码之一,代码表在DataStructureDefinition数据结构定义中作为Dimension维度的Representation表示而指定的。度量维度值能取的实际值不许是一个有效的表示,此表示依据与MeasureDimensionValue度量维度值相关的(+valueFor)ConceptScheme概念组中的Concept概念指定。

ObservationValue观测值可编码-这里指CodedObservation编码的观测-或者也可不编码-这里指UncodedObservation未编码的观测。

GroupKey是Key的子节点，与SeriesKey同维度，但定义SeriesKey的KeyValues关键值的子集。它的子维度结构在与GroupKey同id所认证的DataStructureDefinition数据结构定义的GroupDimensionDescriptor组维度描述中定义。Id确定组的“类型”，同时GroupKey的目的是记录在这个组平面中包含的一个或多个AttributeValue属性值。当GroupDimensionDescriptor组维度描述与DataStructureDefinition数据结构定义中的GroupRelationship组相关性相关时，便出现GroupKey。在一个DataSet数据集中可以存在多种组。如果在DataStructureDefinition数据结构定义中Group组的DimensionRelationship维度相关性相关，则在SeriesKey或Observation观测中AttributeValue属性值将记录有关维度。

用这种方法，DataSet数据集、SeriesKey系列关键、GroupKey组关键和Observation观测都可有零个或多个AttributeValue属性值，属性值定义了关联对象的某些元数据。允许的Concepts概念和这些元数据相关（附属）的对象均在DataStructureDefinition数据结构定义中定义。

AttributeValue属性值与关联的对象类型（DataSet, SeriesKey, Observation）连接。

7.4.3.2 定义

定义见表16。

表16 定义

类	特性	描述
DataSet	抽象类 子级 GenericDataSet StructureSpecificDataSet GenericTime SeriesDataSet StructureSpecificTime SeriesDataSet	数据的有组织合集。
	reportingBegin	在已知的时间区间系统中的特定的时间段，用于确定报告的起始阶段。
	reportingEnd	在已知的时间区间系统中的特定的时间段，用于确定报告的结束阶段。
	dataExtractionDate	确定日期和时间的特定的时间段，在此期间数据从数据源提取。
	validFrom	确定在数据集中信息有效的开始时间。
	validTo	确定在数据集中信息有效的结束时间。
	publicationYear	指定数据或元数据的发布年，依据有效的供应协议内容。
	publicationPeriod	指定数据或元数据的发布周期，依据有效的供应协议内容。

	setId	提供给数据集一个标识。
	action	定义接收系统的动作(更新、附加、删除)。
	describedBy	关联数据流定义, 并且由此将 Data Structure Definition 数据结构定义与数据集关联。
	+structuredBy	关联定义 Data Set 数据集结构的 Data Structure Definition 数据结构定义。注意 Data Structure Definition 数据结构定义与和 Dataflow Definition 数据流定义关联(非强制)的是同一个。
	+publishedBy	关联记录/发布数据的 Data Provider 数据提供者。
	+attachedAttribute	关联与 Data Set 数据集相关的 Attribute Values 属性值。
GenericDataSet		数据格式结构, 可控制与任何 Data Structure Definition 数据结构定义相一致的数据。
StructureSpecific DataSet		数据格式结构, 包含与某个特定 Data Structure Definition 数据结构定义相一致的数据。
GenericTimeseries DataSet		数据格式结构, 可控制与任何 Data Structure Definition 数据结构定义相一致的时间序列数据。
StructureSpecific TimeseriesDataSet		数据格式结构, 包含与某个特定 Data Structure Definition 数据结构定义相一致的时间序列数据。
Key	抽象级 子级 SeriesKey GroupKey	包含维度值的交叉内容, 维度唯一确定一次 Observation 观测。
	keyValues	与组成 Key 关键的单个的 Key Values 关键值关联。
	+attachedAttribute	与 Series Key 系列关键或 Group Key 组关键相关的 Attribute Values 属性值的关联。
KeyValue	抽象级 子级 MeasureKeyValue	关键组件的值, 如在 Data Structure Definition 数据结构定义的 Dimension Descriptor 维度描述中的实际 Dimension 维度

	TimeKeyValue CodedKeyValue UncodedKeyValue	值。
	+valueFor	在 Data Structure Definition 数据结构定义中与关键组件的关联, 使得 Key Value 关键值是有效的表示。 注意这只是概念上的关联, 因为关键组件在数据集中有明确的识别。
MeasureKeyValue	衍生自 KeyValue	关键 Measure Dimension 度量维度组件的值。取值是与类关联的 Concept 概念。
	+value	与 Concept 概念的关联 注意只是概念上的关联, 显示在 Data Structure Definition 数据结构定义中 Concept 概念必须存在于度量维度关联的 Concept Scheme 概念组中。在实际的 Data Set 数据集中 Concept 概念的值位于 Key Value 关键值中。
TimeKeyValue	衍生自 KeyValue	关键的 Time Dimension 时间维度组件的值。
CodedKeyValue	衍生自 KeyValue	关键的编码组件的值。值是类关联的 Code 代码。
	+value	与代码的关联。 注意只是概念上的关联, 显示在 Data Structure Definition 数据结构定义中 Code 代码必须存在于度量维度关联的 Code list 代码表中。在实际的 Data Set 数据集中 Code 代码的值位于 Key Value 关键值中。
UnCodedKeyValue	衍生自 KeyValue	关键的未编码组件的值。
	value	关键组件的值。
	startTime	仅用于在 Data Structure Definition 数据结构定义中属性的 textFormat 文本格式是 Timespan 时间价格类型时 (在此

		情况下值域是可持续的)。
	+valueFor	将 Dimension 维度、Measure Dimension 度量维或 Time Dimension 时间维关联到 Key Value 关键值, 并由此关联到 Dimension 维度或 Time Dimension 时间维的语义 Concept 概念。
GroupKey	衍生自 Key	Key Values 关键值的集, 包含部分关键, 为描述 Data Attributes 数据属性, 与 Time Series Key 时间系列关键有相同的维度。
	+describedBy	与在 Data Structure Definition 数据结构定义中定义的 Group Dimension Descriptor 组维度描述关联。
SeriesKey	衍生自 Key	包含所有 Key Values 关键值的值交叉内容, 同时与+observation Dimension 观测维度的 Key Value 关键值一起唯一确定一次 Observation 观测。
	+describedBy	与在 Data Structure Definition 数据结构定义中定义的 Dimension Descriptor 维度描述关联。
Observation		包含关键的 Key Values 关键值语境中, 所观测的现象值。
	+valueFor	与在 Data Structure Definition 数据结构定义中定义的 Primary Measure 基本维度关联。
	+attachedAttribute	与 Observation 观测值相关的 Attribute Values 属性值的关联。
	+observationDimension	与支撑“Dimension at the Observation Level 在观测面的维度”值的 Key Value 关键值的关联。
ObservationValue	抽象类 子级 UncodedObservation CodedObservation	
UncodedObservation	衍生自 ObservationValue	具有一个文本值的观测。
	value	Uncoded Observation 未编码观测

		的值。
CodedObservation	衍生自 ObservationValue	从 Code list 代码表中的代码取值的 Observation 观测。
	+value	与 Observation 观测的值 Code 代码的关联。 注意这是概念上的关联, 以表示在 Data Structure Definition 数据结构定义中 Code 代码必须存在于与 Primary Measure 基本度量或 Concept of the Measure Dimension 度量维度的概念相关的代码表中。在实际 Data Set 数据集中 Code 代码值位于 Observation 观测值中。
AttributeValue	抽象类 子级 UncodedAttributeValue CodedAttributeValue	属性的值, 例如在结构 (如 Data Structure Definition 数据结构定义) 中的实际 Coded Attribute 代码属性或 Uncoded Attribute 未编码属性。
	value	属性的值。
	+valueFor	与在 Data Structure Definition 数据结构定义中定义的数据 Attribute 数据属性关联。注意这是概念性的关联, 因在数据集中已对 Concept 概念有明确地识别。
UncodedAttribute Value	衍生自 AttributeValue	有文本值的属性值。
	startTime	仅用于在 Data Structure Definition 数据结构定义中属性的 textFormat 文本格式是 Timespan 时间价格类型时 (在此情况下值域是可持续的)。
CodedAttribute Value	衍生自 AttributeValue	一个从 Code list 代码表中的 Code 代码值取值的属性。
	+value	与 Attribute Value 属性值的值 Code 代码的关联。 注意这是概念上的关联, 以表示在 Data Structure Definition 数据结构定义中 Code 代码必须存在于与 Data Attribute 数据属性相关

		的 Code list 代码表中。在实际 Data Set 数据集中 Code 代码值位于 Attribute Value 属性值中。
--	--	---

8 立方体

8.1 背景

一些统计系统基于“多维框架”结构建立数据视图。其实，Cube（立方体）结构是一个n维的对象，在其中每一个维度的值均来源于一个多层的代码表。这种方体结构系统的效用是使得对每一个维度每个分层平面都能实现“上卷”或“下钻”，从而规定所需要给出数据“视图”的颗粒度间隔水平-有些维度或许需要上翻，其他或许需要下钻。这种系统实现了数据的动态观测，运用对上卷式的维度定位总值。例如，独立国家或许上卷到一个经济区，如EU欧盟，或一个地理区域如欧洲，同时另一个维度，如“公路类型”或许下钻到它的更低层面。结果的估量（如“事故的数量”）将依据公路的特定类型对每个独立的国家形成值的合计。

这种cube方体系统不依靠简单的代码表，而是需要多的层代码集。

8.2 在信息模型中对 cube 的支持

运用Data Structure Definition数据结构定义架构（每个维度值如果是编码的，则取自一维的代码表）记录的数据可被cube多维定义描述并用cube数据识别系统控件处理。SDMX-IM通过如下方式支持cube的定义：

- HierarchicalCodelist 多层代码表定义代码的分层（通常是复杂的分层）；
- 如有需求，StructureSet 结构集可以
 - a) 将描述 cube 的 DataStructureDefinition 数据结构定义分组；
 - b) 提供绘图机制，在用于 DataStructureDefinition 数据结构定义的一维代码表中的代码与 HierarchicalCodelist 层级代码表之间，HierarchicalCodelist 层级代码表在 DataStructureDefinition 数据结构定义中不使用。

9 元数据结构定义和元数据集

9.1 背景

SDMX元模型允许元数据：

- a) 交换，不需要嵌入所描述的对象中；
- b) 与所描述的对象分开存储，但可链接（例如，一个机构有元数据仓库，可根据可接近元数据属于的对象的系统或服务产生的元数据需要进行元数据的发布。运用网络发布很常见，可以通过在元数据关联的对象边设置“信息”按钮就能实现附加元数据的查询（甚至下载））；
- c) 可检索（例如：一个注册服务可以处理元数据报表和提取结构信息，信息内容是允许元数据以用户可检索的方式编成目录）；
- d) 可根据定义的结构记录发布。

为了实现这些，依据以下结构建模：

——元数据结构定义有如下组件：

- a) 对象类型，元数据所关联（附属的）对象；
- b) 相互的组件包含于元数据关联的对象类型的唯一关键 key；

- c) 包含可关联多种对象类型的元数据属性（这些属性可在一个层级中构建）的报表结构，连同可能的约束条件（例如，关联到包含属性有效值的代码表，记录在元数据集中）；
——元数据集，包含记录的元数据。

9.2 继承

9.2.1 引言

模型这个层面的构念由于是与Data Structure Definition Structure数据结构定义架构一起的，因此许多构念来源于SDMX的Base layer基础层。因此，有必要学习继承和关系图表来理解每个独立功能包的功能。以下图表显示了与MetadataStructureDefinition元数据结构定义和MetadataSet元数据集相关的类的整体继承树的情况。

在MetadataStructureDefinition元数据结构定义功能包中极少数附加类不从SDMX Base的类中衍生。换句话说，SDMX Base给出了这个子模型的大部分关联性和属性结构。本章的关系图表清楚地展示了从SDMX Base衍生的关联性。（查看附录A中用于描述的图解注释）。应该注意，用于MetadataStructureDefinition元数据结构定义的SDMX基础结构和用于DataStructureDefinition数据结构定义的是相同的，因此，纵然用法有轻微的差别，但定义MetadataStructureDefinition元数据结构定义的潜在方法与用于定义DataStructureDefinition数据结构定义的方法是相似的。

9.2.2 类图-继承

元数据结构定义继承类图见图25。

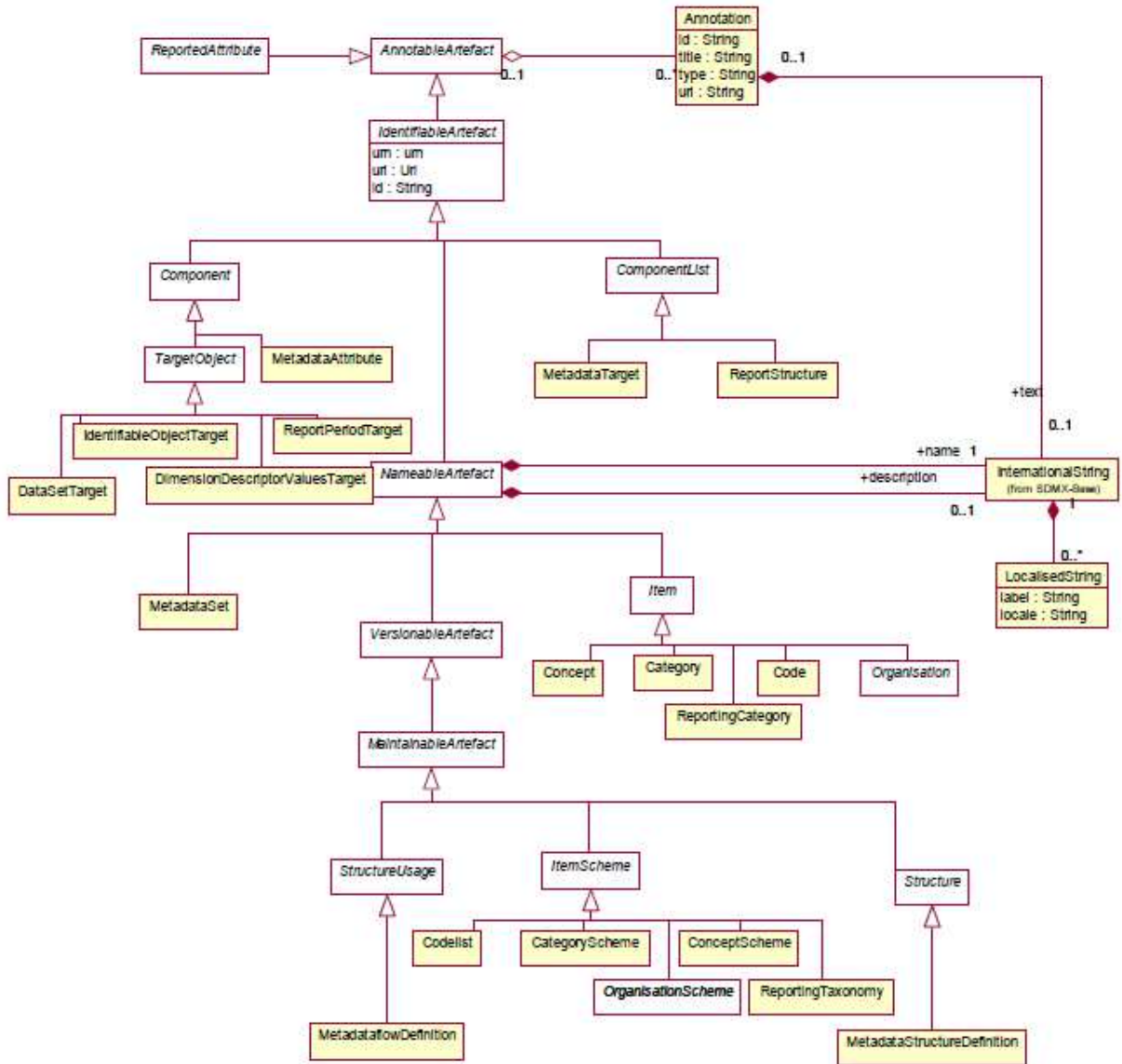


图25 元数据结构定义继承类图

9.2.3 图解

对本章关系图的理解是很重要的，以便能分辨从抽象类中衍生出的具体类。

SDMX元模型的本部分内的实际类均衍生自MaintainableArtefact，同时由Maintenance Agencies维护代理负责维护。它们是：

- StructureUsage (实际类是 MetadataflowDefinition)；
- Structure (实际类是 MetadataStructureDefinition)。

这些类同样继承了 IdentifiableArtefact、NameableArtefact、VersionableArtefact 的身份和版本信息。

Structure 包含几个组件清单。从 ComponentList 组件清单中衍生并存在于其中的具体类是 MetadataStructureDefinition 元数据结构定义的子组件。它们是：

- MetadataTarget；

——ReportStructure.

ComponentList 组件清单包含ComponentS组件。从Component组件中衍生的类有:

——Sub Classes of TargetObject;

——MetadataAttribute.

9.3 元数据结构定义

9.3.1 引言

本章剩下的图表和介绍将展现这些具体的类是如何关联并支持功能需求。

9.3.2 已描述的结构

MetadataStructureDefinition元数据结构定义运用了如下ItemScheme架构,既作为模型中明确的实际类,又作为包含TargetObject值域的可能清单。

——CategoryScheme;

——ConceptScheme;

——CodeList;

——OrganisationScheme;

——Reporting Taxonomy.

9.3.3 类图-关系

元数据结构定义关系类图见图26。

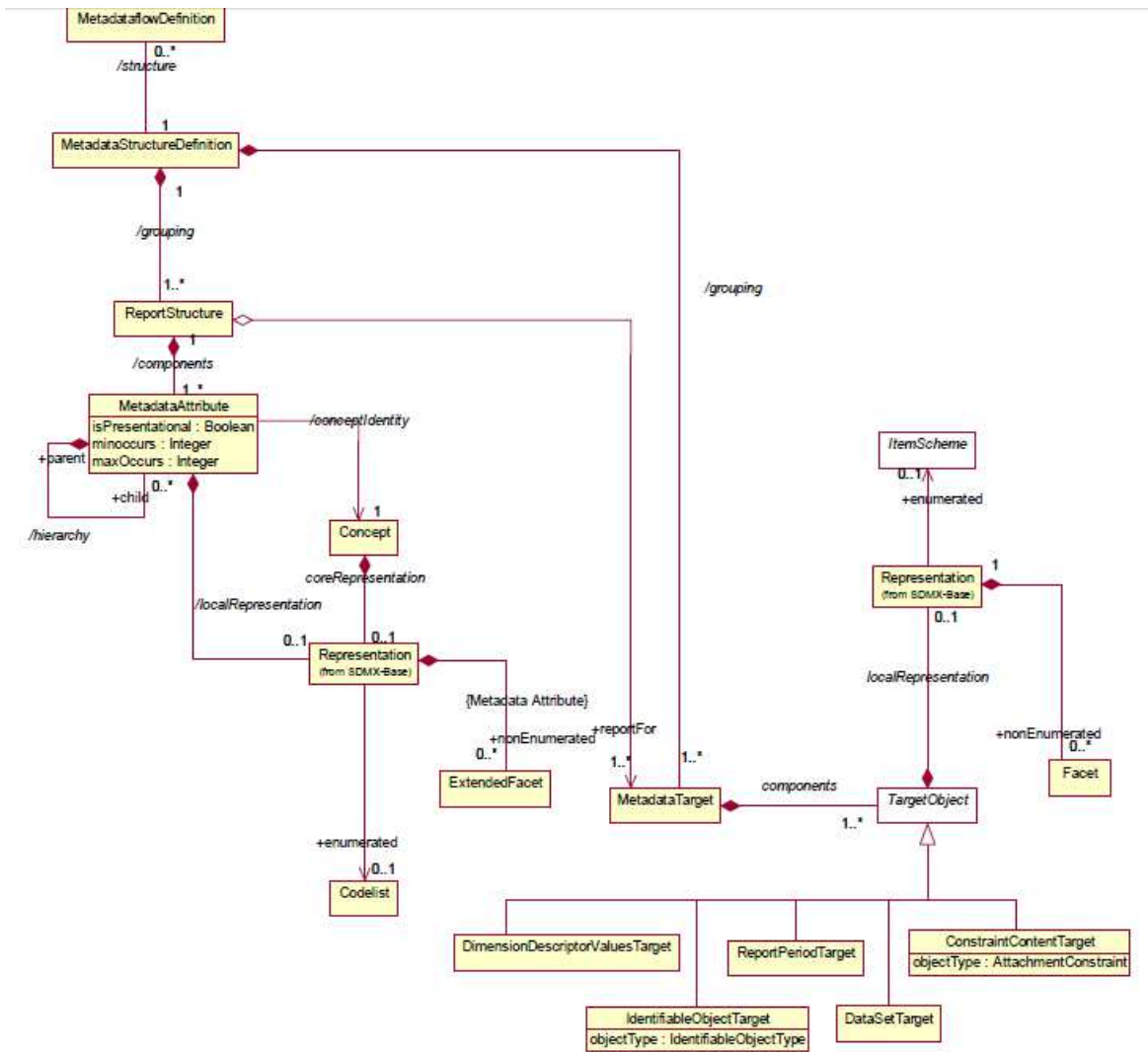


图26 元数据结构定义关系类图

9.3.4 图解

9.3.4.1 叙述

简单的说，MetadataStructureDefinition (MSD) 元数据结构定义定义了如下内容：

- MetadataTarget 元数据目标，定义组件 (TargetObject 目标对象) 和对 MetadataStructureDefinition 元数据结构定义有效的 Representation 表示，同时这是一个或多个 ReportStructure 记录架构的元数据目标对象；
- ReportStructureS 记录结构，包含内容是 MetadataAttribute 元数据属性，与在 MetadataTargetS 参考元数据目标中识别的目标类型相关联，同时包含属性的分层结构。
- MetadataTarget 元数据目标包含一个或多个 TargetObjectS 目标对象。
- TargetObjectS 目标对象的结合可以识别元数据在 MetadataSet 元数据集中所附属的特定对象类型。

TargetObject 目标对象是以下之一：

- DimensionDescriptorValuesTarget 维度描述值目标-允许在 MetadataSet 元数据集中整体或部分的关键 (如在数据集中的应用一样) 规定作为目标对象而指定；

- IdentifiableObjectTarget 可识别对象目标-定义特定的对象类型，可以是任意的 IdentifiableArtefact 可识别产成品；
 - DataSetTarget 数据集目标-规定目标对象是一个 DataSet 数据集；
 - ReportPeriodTarget 报告期目标-规定报告期必须存在于 MetadataSet 元数据集中；
 - ConstraintContentTarget 约束内容目标-规定目标对象是 AttachmentConstraint 附属约束条件的内容。集数据集或元数据集的部分由 AttachmentConstraint 附属约束的内容识别。
 - 当记录在 MetadataSet 元数据集中时，TargetObject 目标对象的有效内容在 Representation 表示中定义。内容可以是可枚举的表示（即参照 ItemScheme 项目方案的一个子级-有 Codelist 代码表、ConceptScheme 概念列表、OrganisationScheme 组织列表、CategoryScheme 分类列表、或 ReportingTaxonomy 报告期分类）或不可枚举的。
 - 因此，可以对相关的目标类型的离散集定义单一的 MetadataStructureDefinition 元数据结构定义。例如，可构造单一的定义以规定与 Data Structure Definition 数据结构定义的任何部分关联的元数据，或者关联与品质元数据（如数据提供者和（数家）分类）报告相关的任何加工品。在 MetadataSet 元数据集中，MetadataTarget 元数据目标制定元数据可关联的特定目标类型的身份属性。例如，在一个 DataStructureDefiniton 数据结构定义中 MetadataTarget 元数据目标或许是 Dimension 维度，因此 TargetObjects 目标对象可唯一识别一个 Dimension 维度。这项包括 DataStructureDefinition 数据结构定义和 Dimension 维度两个内容（两者均是 IdentifiableArtefact 可识别产成品，都使用 IdentitifiableObjectTarget 可识别对象目标）因为需要两个的 TargetObjects 目标对象以识别唯一的 Dimension 维度。
 - ReportStructure 报表结构 MetadataAttributes 元数据属性集组成-这些内容可作为一个层定义。每个 MetadataAttribute 元数据属性识别一个在 MetadataSet 元数据集中公布或分发的 Concept 概念，MetadataSet (/conceptIdentity 概念身份) 元数据集使用所在的 MetadataStructureDefinition 元数据结构定义。在同一个 ReportStructure 报表结构中，不同的 MetadataAttributes 元数据属性可使用来自不同 ConceptSchemes 概念列表的 Concepts 概念。注意一个 MetadataAttribute 元数据属性不与在 MetadataStructureDefinition 元数据结构定义中定义其角色的 Concept 概念相连（即 MetadataAttribute 元数据属性不扮演一个角色）。
 - MetadataAttribute 元数据属性可以规定为多次事件和 / 或定义为强制性的（minOccurs=1 or more 至少发生一次或多次）或条件性的（minOccurs=0 最小发生次数为 0）。一个分层的 ReportStructure 报表结构可由一个 MetadataAttribute 元数据属性的特定分层定义。
 - ReportStructure 报表结构与一个或多个 MetadataTargets 元数据目标关联，元数据目标规定了当 MetadataSet 元数据集需记录目标对象时，在 ReportStructure 报表结构中 MetadataAttributes 元数据属性指定的对象。
- 从中可以看出 MetadataAttribute 元数据结构属性关联的目标类型的规格是间接的：
- MetadataAttributes 元数据属性是在一个或多个 MetadataTarget 元数据目标关联的 ReportStructure 报表结构中定义的，同时真实的对象由 MetadataTarget 元数据目标关联的 TargetObjects 目标对象定义。如此形成了灵活的机制，使得实际的对象类型不需要在模型中用固定的术语定义，但在 MetadataStructureDefinition 元数据结构定义中可动态定义，如同在 DataStructureDefinition 数据结构定义中的数据观测值的相关关

键点定义一样。用这种方法，MetadataStructureDefinition 元数据结构定义通常定义 MetadataAttributes 元数据属性的任意集合所关联的任意对象类型集。

- 每个 MetadataAttribute 元数据属性可以有一个特定的 Representation 表示（/使用 localRepresentation 关联）。如果没有在 MetadataStructureDefinition 元数据结构定义中指定，那么 MetadataStructureDefinition 表示从 Concept 概念的定义中提取（coreRepresentation 核心表示关联）。
- Representation 表示的各种类型定义可在基本构架说明中找到。注意如果 Representation 表示是不可枚举的，那么就与 ExtendedFacet 扩展面关联（它对应 xhtml 允许作为一个 FacetValueType 面值类型）；如果 Representation 表示是可枚举的，那么必须使用 Codelist 代码表。
- MetadataStructureDefinition 元数据结构定义与 MetadataflowDefiniton 元数据流定义连接。除衍生自 Base classed 基础类的属性外，MetadataflowDefiniton 元数据流定义不包括任何属性。

9.3.4.2 定义

定义见表17。

表17 定义

类	特性	描述
StructureUsage		参见“SDMX Base”。
Metadataflow Definition	衍生自 StructureUsage	元数据流的抽象概念（即没有任何元数据的结构），由提供者依据不同的参照期间提供。
	/structure	关联一个 Metadata Structure Definition 元数据结构定义。
MetadataStructure Definition		元数据概念的合集，当用于搜集或分发参考元数据时的结构和用法。
	/grouping	与 Metadata Target 元数据目标或 Report Structure 报表结构的关联。
MetadataTarget	衍生自 ComponentList	定义与元数据相关的对象类型的关键值的组件集。
	/components	关联 Target Object 目标对象组件，定义 Metadata Target 元数据目标的关键值。
TargetObject	抽象类 子级 DimensionDescriptorValues Target IdentifiableObjectTarget DataSetTarget	

	ReportPeriodTarget	
	/localRepresentation	关联 Target Object 目标对象的 Representation 表示, 在 Metadata Set 元数据集中定义对象时必须重视。可以是可枚举的或不可枚举的。
DimensionDescriptor ValuesTarget	衍生自 TargetObject	目标对象是数据序列的关键。
IdentifiableObject Target	衍生自 TargetObject	目标对象是指定的对象类型。
	objectType	识别对象类型。
DataSetTarget	衍生自 TargetObject	目标对象是 Data Set 数据集。
ReportPeriodTarget	衍生自 TargetObject	目标是报告周期。注意不是用于描述对象的使用, 而是为元数据报告提供唯一的元数据关键值。与特殊对象关联的元数据报告可时间多样, 而这个时间识别组件可用于解疑报告, 很像在数据序列中时间维度对观测值的澄清。
ConstraintTarget	衍生自 TargetObject	目标对象是数据或参考元数据, 在 Attachment Constraint 附件约束条件中识别。
ReportStructure	衍生自 ComponentList	定义组成报告 Metadata Attributes 元数据属性的概念集。
	/components	与 Report Structure 报表结构相关的 Metadata Attributes 元数据属性的关联。
	+reportFor	关联使用 Report Structure 报表结构的 Metadata Targets 元数据目标。
MetadataAttribute		识别在 Metadata Set 元数据集中可能记录值的 Concept 概念。
	/hierarchy	与一个或多个子 Metadata Attribute 元数据属性的关联。
	/conceptIdentity	与定义属性语义的概念的关联。
	isPresentational	出于结构目的 Metadata Attribute 元数据属性的表现 (即存在子属性), 在 Metadata Set 元数据集中使用此 Report Structure 报表结构时属性没有

		报告值。
	minOccurs maxOccurs	指定 Metadata Attribute 元数据属性在 Metadata Report 元数据报表中的事件数量。
ConceptUsage		作为 Metadata Attribute 元数据属性的 Concept 概念的使用。
	concept	在 ConceptScheme 概念列表中与 Concept 概念的关联。
	/localRepresentation	关联可重写由 Concept 概念指定的任何核心表示的 Representation 表示。
Representation		Metadata Attribute 元数据属性的表示。

9.4 元数据集

9.4.1 类图

元数据集关系类图见图27。

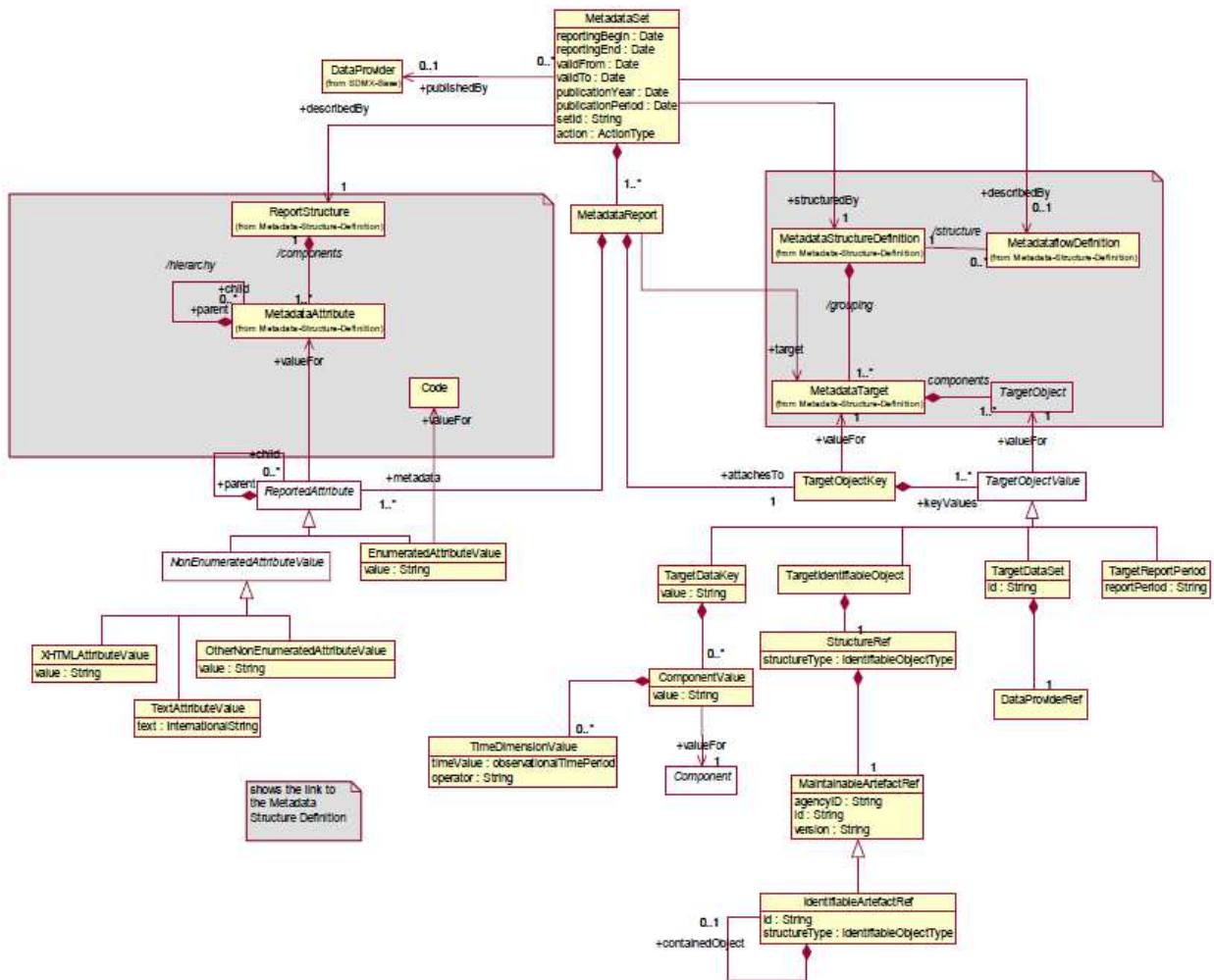


图27 元数据集关系类图

9.4.2 图解

9.4.2.1 叙述

注意MetadataSet元数据集必须符合与MetadataStructureDefinition元数据流定义相关联的MetadataStructureDefinition元数据结构定义,在此环境下MetadataSet元数据集是“元数据的实例”。同时,模型显示了MetadataStructureDefinition元数据结构定义的类的关联性,这是出于概念上的目的而显示的与MetadataflowDefinition元数据结构定义的连接。在实际的MetadataSet元数据集中,用于交换当然必须参照MetadataflowDefinition元数据结构定义和ReportStructure报表结构,同时可选的MetadataflowDefinition元数据流定义,但是MetadataflowDefinition元数据结构定义不是必须与元数据交换的。因此,MetadataflowDefinition元数据结构定义的类用灰色区域显示,因为它们不是MetadataSet元数据集本身的一部分。

作为DataProvider数据提供者角色的组织对一个或多个MetadataSet元数据集负责。

一个MetadataSet元数据集由一个或多个MetadataReport元数据报表组成,每个必须使用相同的ReportStructure报表结构。元数据参考在MetadataStructureDefinition元数据结构定义中定义的MetadataTarget元数据目标,同时包含TargetObjectKey目标对象关键和ReportedAttributes报表属性。

已认证的ReportStructure报表结构指定哪些MetadataAttributes元数据属性作为ReportedAttributes报表属性。已认证的MetadataTarget元数据目标指定TargetObjectKey目标对象关键的内容,即指定需要识别对象的信息,这些对象是ReportedAttributes报表属性所记录的。

TargetObjectValue目标对象值可是以下之一:

——TargetObjectValue 目标数据关键-包括:

- 一个SeriesKey 系列关键(维度值的集)
- 一个SeriesKey 系列关键加一个或多个TimeDimension (TimeDimensionValue) 时间维度值(给出时间范围)
- 一个或多个TimeDimension 时间维度值

——TargetIdentifiableObject 目标可认证对象-识别任何可识别的对象(包括 Maintainable 可维护和 Identifiable 可识别对象)

——TargetDataSet 目标数据集-识别 DataSet 数据集

——TargetReportPeriod 目标报告周期-指定 Report 报表的报告周期

——对于 ReportedAttribute 报表属性简单的文本值使用的是 ReportedAttribute 报表属性子级的 NonEnumeratedAttributeValue 不可枚举属性值,同时编码值使用子级 EnumeratedAttributeValue 可枚举属性值。

NonEnumeratedAttributeValue 不可枚举属性值有:

——XHTMLAttributeValueXHTML 属性值-内容为 XHTML

——TextAttributeValue 文本属性值-内容是文本的,或许包括多语言 text 文本

——OtherNonEnumeratedAttributeValue 其他不可枚举属性值-内容是一个字符串值,必须符合在 MetadataStructureDefinition 元数据结构定义中与 ReportStructure 报表结构相关的 MetadataAttribute 元数据属性指定的 Representation 表示

——可枚举属性值包括代码的值,代码作为在 MetadataStructureDefinition 元数据结构定

义中与 ReportStructure 报表结构相关的, MetadataAttribute 元数据属性的 Representation 表示。

9.4.2.2 定义

定义见表18。

表18 定义

类	特性	描述
MetadataSet		任意元数据的有机合集。
	reportingBegin	在已知的时间周期系统中识别报告开始时期的指定时间周期。
	reportingEnd	在已知的时间周期系统中识别报告结束时期的指定时间周期。
	dataExtractionDate	识别数据从数据源提取的时间和日期的指定时间周期。
	validFrom	表明在数据集中信息有效的开始时间。
	validTo	标明在元数据集中信息有效的结束时间。
	publicationYear	指定数据或元数据的发布年, 依据有效的供应协议内容。
	publicationPeriod	指定数据或元数据的发布周期, 依据有效的供应协议内容。
	setId	提供给元数据集一个标识。
	action	定义接收系统的动作(更新、附加、删除)。
	+describedBy	关联 Metadataflow Definition 元数据流定义与 Metadata Set 元数据集。
	+structuredBy	关联定义 Metadata Set 元数据集结构的 Metadata Structure Definition 元数据结构定义。注意 Metadata Structure Definition 元数据结构定义与和 Metadataflow Definition 数据流定义关联(非强制)的是同一个。
	+publishedBy	关联记录/发布数据的 Data Provider 数据提供者。
	+describedBy	参考 Report Structure 报表结构。
MetadataReport		在 Metadata Structure Definition 元数据结构定义的 Report Structure 报表结构中定

		义的 Metadata Attributes 元数据属性的值集。
	+attachesTo	关联对象关键值与要附属的元数据。
	+target	关联定义元数据相关的目标对象的 Metadata Target 元数据目标。
	+metadata	关联由 Target Object Key 目标对象关键识别的对象的 Reported Attribute 报告属性值。
TargetObjectKey		识别元数据附属的对象关键值。
	+valueFor	关联识别 Target Object Key 目标对象关键的对象类型和组件结构的 Metadata Target 元数据目标。 注意这是概念性的关联，显示与 MSD 构建的连接。
	+keyValues	关联 Target Object Key 目标对象关键的 Target Object Values 目标对象值。
TargetObjectValue	抽象类 子级 TargetDataKey TargetIdentifiableObject TargetDataSet TargetReportPeriod	在 Metadata Structure Definition 元数据结构定义的 Metadata Target 元数据目标中指定的单个对象类型的关键值。
	+valueFor	关联提供值的 Target Object 目标对象。 注意这是概念性的关联，显示与 MSD 构建的连接。
TargetDataKey	衍生自 TargetObjectValue	组成数据或元数据关键的组件认证和值。
ComponentValue		整体包括形成数据关键的组件认证和值。
value		关键值。
	+valueFor	关联值所声明的 Component 组件。
TimeDimensionValue		包括 Time Dimension 时间维和值的认证。
TargetIdentifiableObject	衍生自 TargetObjectValue	指定可认证对象的认证。
StructureRef		包括可认证对象的认证。

	structureType	目标对象的对象类型。
Maintainable ArtefactRef Identifiable ArtefactRef		用于标示符构架的目标对象的识别，即代理 ID, id, 对附加可维护对象、可认证对象的版本，特定 id。
	+containedObject	与在 Identifiable Objects 可识别对象层（如在 Process Step 处理步骤中的 Transition 转换）的包含对象的关联。
TargetDataSet	衍生自 TargetObjectValue	包含 Data Set 数据集的认证。
TargetReportPeriod	衍生自 TargetObjectValue	包含 Metadata Report 元数据报告覆盖的周期。
ReportedAttribute	抽象类 子级 NonEnumeratedAttributeValue EnumeratedAttributeValue	Metadata Attribute 元数据属性的值。
	+valueFor	关联在 Metadata Structure Definition 元数据结构定义中的 Metadata Attribute 元数据属性，识别用于 Reported Attribute 报告属性的 Concept 概念和允许的 Representation 表示 注意这是概念性的关联，显示与 MSD 构建的连接。Reported Attribute 报告属性的语法在某些表格中依据 Metadata Attribute 元数据属性的 id 设置。
	+child	关联与在 Report Structure 报告结构中依据 Metadata Attribute 元数据属性定义的层意志的子 Reported Attribute 报告属性，因此为一种 Reported Attribute 报告属性。
NonEnumerated AttributeValue	衍生自 ReportedAttribute 子级 XHTMLAttributeValue TextAttributeValue OtherNonEnumerated	报告属性的内容，文本形式。

	AttributeValue	
XHTMLAttributeValue		包含 XHTML。
	value	XHTML 的字符串值。
TextAttributeValue		Reported Attribute 报告属性值，内容为可人读的文本。
	text	字符串值是文本。可存在多语言版本。
OtherNonEnumerated AttributeValue		Reported Attribute 报告属性值，内容不是可人读的文本。
	value	在格式上一致的文本字符串，在 Metadata Attribute 元数据属性的 Representation 表示中定义，故为 Reported Attribute 报告属性。
EnumeratedAttribute Value	衍生自 MetadataAttributeValue	Reported Attribute 报告属性的内容，从 Code list 代码表的一个 Code 代码中获得。
	value	Reported Attribute 报告属性的 Code 代码值。
	+value	关联在 Metadata Attribute 元数据属性的 Representation 表述中指定的 Code list 代码表中的一个 Code 代码，因此取这个 Reported Attribute 报告属性为值。 注意显示的是与值项的概念性连接。实际上，值本身将包含在 Enumerated Attribute Value 可枚举的属性值中。

10 层级代码方案

10.1 范围

本章结构定义中说明的 CodeList 支持 Codes 的简单层次，约束全部仅有一个父代码的子代码。同时，这个结构对支持 KeyFamily 和 MetadataStructureDefinition 的需要是有用的，它不足以支持更多的基于代码方案（如分类方案）的代码之间的复杂关联。通常，KeyFamily 里使用的 CodeList 来自更复杂的代码方案。访问这样的代码方案有助于应用程序，如 OLAP 应用程序，与 KeyFamily 里使用的简单 CodeList 相比，它能给出更多的数据视图。

注意一个层级代码表不必是一个平衡树。有平衡树的等级是预定义的和固定的（即一个等级总是有同样的代码集，并且所有的代码与其它代码都有固定的父母子女关系）。统计分类是平衡树的一个例子，且平衡分层的支持是一个子集并且是层级代码表的特例。

层级代码方案的基本特征是：

- a) 子代码可以有多个父代码；
- b) 可以有多个代码没有父代码（即，多个“根节点”）；
- c) 在代码之间的关联项里，可能定义了许多层次（或“视图”）。每一个层次提供报告、分析或数据传输。
- d) 层次的等级可以是显式的也可以是隐式的：（即他们在编码结构中只存在父代码/子代码的关系）。

10.2 继承

10.2.1 类图

代码集（Code Set）的继承类图见图28。

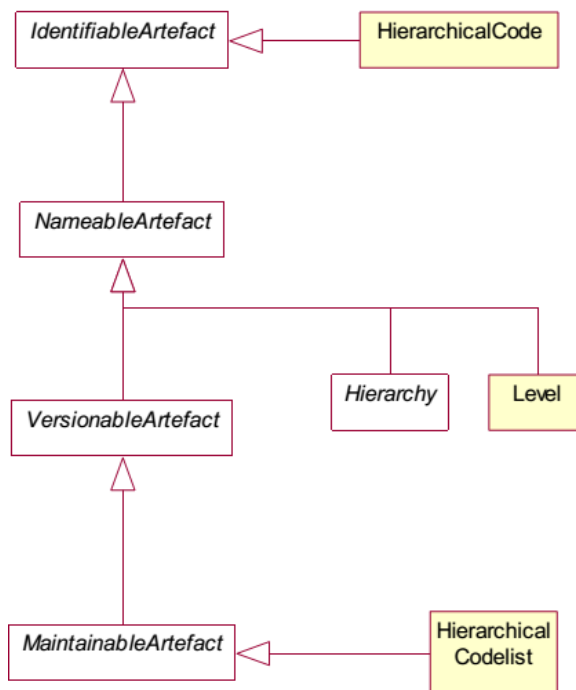


图28 代码集（Code Set）的继承类图

10.2.2 图解

10.2.2.1 叙述

HierarchicalCodeScheme 继承自 *ItemScheme*，因此是一个有身份标识、版本和维护机构的 *MaintainableArtefact*。*CodeAssociation* 继承自 *CodeMap*（见第11章），因此是一个 *VersionableArtefact*。*Hierarchy* 直接继承自 *VersionableArtefact*。因此两个都有身份标识和版本。*Level* 是一个 *IdentifiableArtefact*，因此有一个ID、多语言名称和多语言说明。

理解组成 *HierarchicalCodeScheme* 的 *Code* 不是它们自己包含在方案里是重要的，它们引用自方案且在一个或多个 *CodeList* 里维护。下面的关系类图解释对此进行了说明。

10.2.2.2 定义

各种类、属性和关联的定义出现在下面的关系节里。

10.3 关系

10.3.1 类图

层级代码方案的关系类见图29。

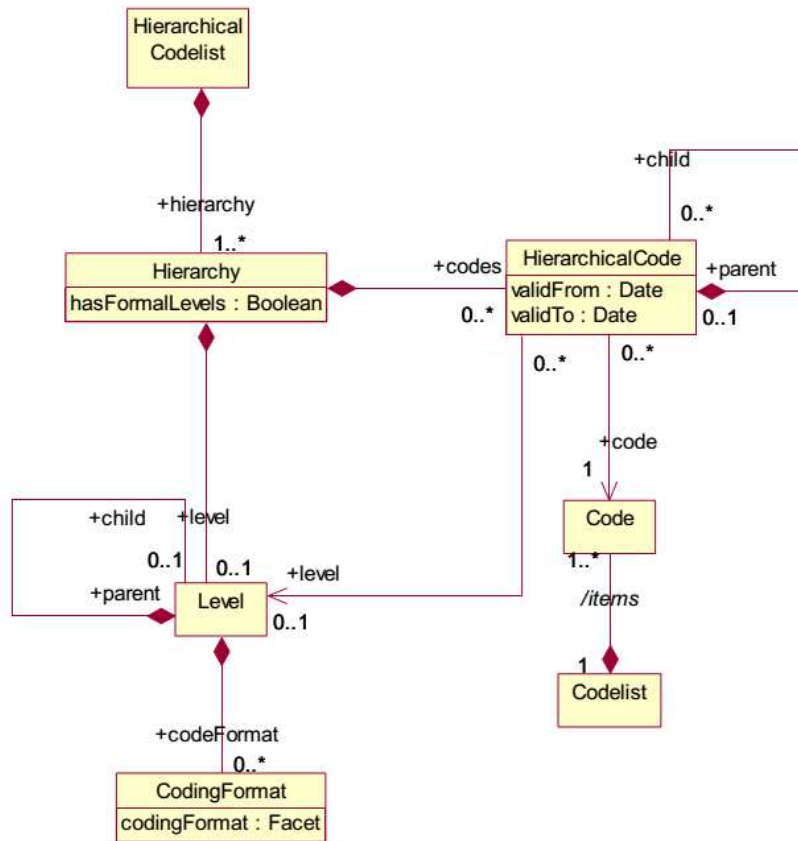


图29 层级代码方案的关系类图

10.3.2 图解

10.3.2.1 叙述

HierarchicalCodeScheme里的关联和关联的可导航性受到约束：以确保HierarchicalCodeScheme基本功能一致性的通用实现。至关重要的是：

- a) HierarchicalCodeScheme是代码的说明，Code包含方案和一个或多个层级方案里的代码的结构说明；
- b) HierarchicalCodeScheme里的代码本身不是方案的一部分，一定程度上它们是对在一个或多个外部CodeList里代码的引用；
- c) 这些代码可能是一个或多个Hierarchy的一部分，也可能是一个或多个为了提供结构给HierarchicalCodeScheme的CodeComposition的一部分；
- d) CodeAssociation里指定了任意两代码之间的关联，该关联局限于标识代码和它的父代码；一个层级可有正式的级（hasFormalLevels="true"为真）。然而，即使hasFormalLevels="false"为假，层级仍可有一个或多个关联层级以便记录有关层级代码的信息。

如果hasFormalLevels="false"为假，“基于值”的层级由无正式层级的代码层组成。如果hasFormalLevels="true"为真，则层是每个层级都在层级代码表中有正式层级的“基于级”，例如

在统计分组中的层。在一个“基于级”的分层中，每个层级代码表与所在的层级连接（级必须与层级代码在同一个层级内）。预期所有根据+parent/+child父/子关联定义的在同一层级的层级代码将连接到相同的层级。注意+level关联在以下情形需要唯一指定，如层级代码在不同的层级（（默认的根据层级代码父/子关联）而得的级比实际在级层（根据Level父/子关联默认的）中的Level）。

注：希望使用对统计分类可接受的模型的组织需确定ID是与Level级关联的数字，在此Levels级是连续地编号的，1级为最高Level级。

Level可以有定义的CodingFormat编码格式信息（例如在此级的编码类型）。

10.3.2.2 定义

定义见表19。

表19 定义

类	特性	描述
HierarchicalCode list	衍生自 MaintainableArtefact	代码的有组织合集，可在设计图中与其他代码一起参与多个父/子关系，根据设计图中的一个或多个层级定义。
	+hierarchy	与层级代码的关联。
Hierarchy	衍生自 NameableArtefact	分类结构，按细节层次排列，从最广义的到最细节的级。
	hasFormalLevels	如果“true”表明层结构按细节层次从最广义的到最细节的级排列。 如果“false”表明层结构内层的各项均无正式级结构。
	+codes	在层级中与最高级层级代码的关联。
	+level	在层中与最高级的关联。
Level	衍生自 NameableArtefact	在“基于级”的层中，描述具备同质编码特性的Codes代码组，同时族中每个Code代码的父在Hierarchy层中的同高度级中。 在“基于值”的层中，描述有关指定嵌套级的HierarchicalCodes层代码信息。
	+codeFormat	与CodingFormat编码格式的关联。
	+child	与级的子级的关联。
CodingFormat		指定层内这个级的代码格式信息，如本级的代码是字符型的、数值型还是字母数值型以及代码长度。
HierarchicalCode		代码参照的层结构。
	validFrom	构想有效的起始日期。
	validTo	构想废止的日期。
	+code	在层中与用于特定点的代码的关联。
	+child	在层中与子代码的关联。
	+level	与依据层级定义的级的所在级的关联。

Code		在层中用于此节点的代码。
	/items	与包含代码的代码表的关联。
Codelist		包含代码的代码表。

11 结构集和映射

11.1 范围

StructureSet允许一个结构里的组件映射到同类型的另一个结构的组件上。该环境下，使用术语“结构(structure)”，包括ItemScheme的类型，Structure的类型，StructureUsage的类型。这些结构中允许映射的组件见表20：

表20 组件映射

结构类型 (Structure Type)	组件类型 (Component type)
Code List	Code
Category Scheme	Category
Concept Scheme	Concept
Data Structure Definition(Key Family)	Dimension, Data Attribute, Measure
Metadata Structure Definition	Identifier Component, Metadata Attribute
Dataflow Definition	Data Structure Definition(Key Family)
Metadataflow Definition	Metadata Structure Definition

StructureSet结构集可包括一个或多个“映射”，同时能定义相关结构（通过+relatedStructure相关结构关联）：这种相关结构聚集了相关的DataStructureDefinitions数据结构定义，MetadataStructureDefinitions元数据结构定义，DataflowDefinintions数据流定义，MetadataflowDefinintions元数据流定义。

11.2 结构集

11.2.1 类图--继承

结构集类见图30。

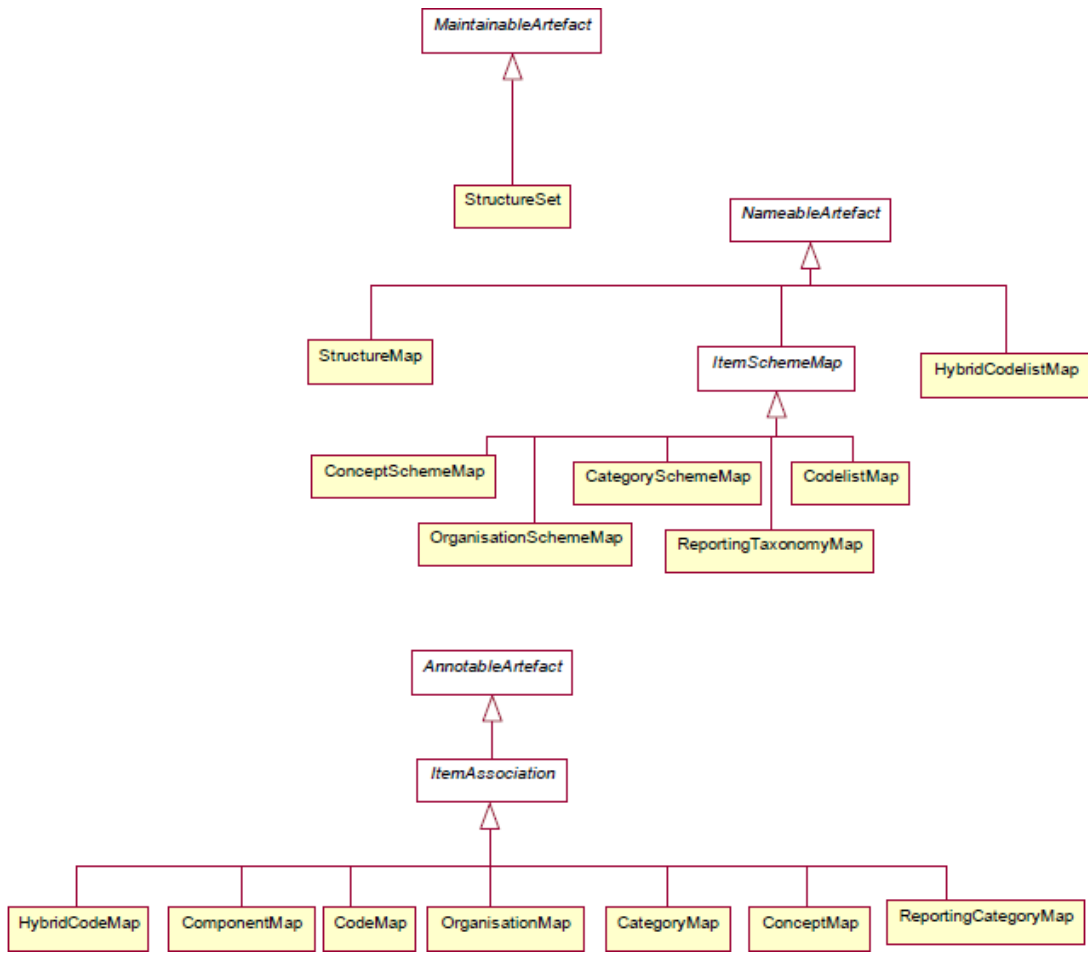


图30 结构集类图

11.2.2 类图—关系

图的说明见图31。

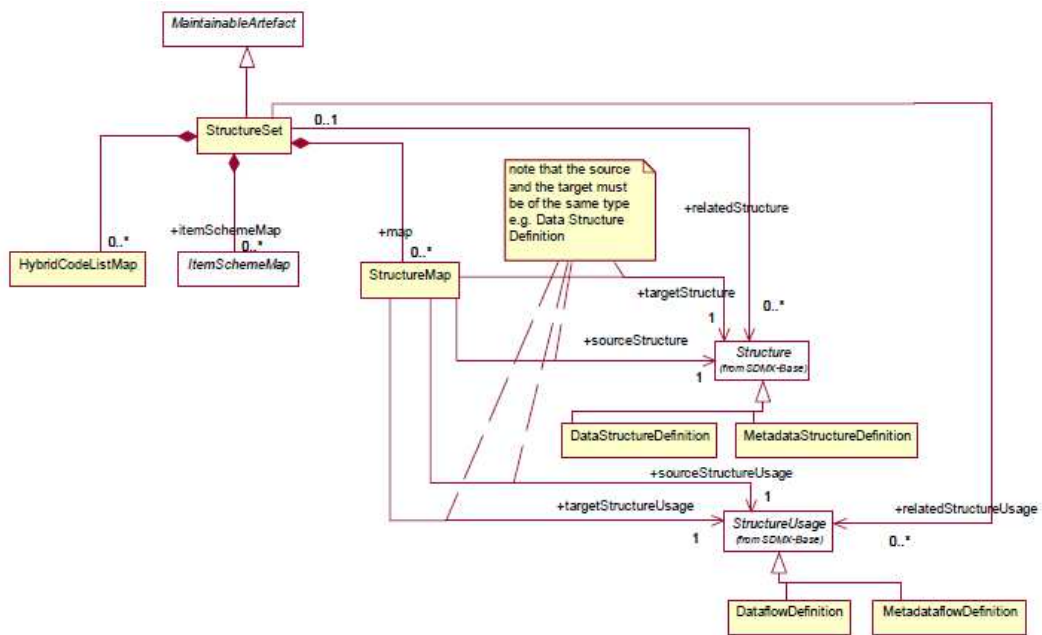


图31 图的说明—关系

11.2.3 图的说明

11.2.3.1 叙述

StructureSet是一个MaintainableArtefact。它可以包含：

- a) Structure和StructureUsage（数据结构定义DataStructureDefinition、元数据结构定义MetadataStructureDefinition、数据流定义DataflowDefinition或元数据流定义MetadataflowDefinition）的实体子类的引用集指出一个语义关系存在于它们之间。例如，可能有共同定义一个立方体的数据结构定义，每一个数据结构定义DataStructureDefinition定义该立方体的一部分。
- b) StructureMaps结构映射集，定义在ComponentMap组件映射中一个结构的哪些组件相当于另一个结构中的哪些组件。
- c) ItemSchemeMaps项目列映射集，定义ItemScheme项目方案的两个具体级间的映射，和这些列表中Items项目的映射，例如两个Codelists代码表中Codes代码的映射。
- d) HybridCodeListMaps混合代码表映射集，定义CodeList代码表和HierarchicalCodeList混合代码表间的映射。

StructureMap结构映射参照两个Structures结构或StructureUsages结构用法。在具体条件下，这些参考是数据结构定义、元数据结构定义、数据流定义或元数据流定义。

11.2.3.2 定义

定义见表21。

表21 定义

类	特性	描述
StructureSet	衍生自 MaintainableArtefact	结构映射图的可维护合集，在源/目标关系中连接组件，在此源和目标组件是语义上的对等关系。

	+relatedStructure	与 Data Structure Definitions 数据结构定义和 Metadata StructureDefinitions 元数据结构定义集的关联。
	+relatedStructureUsage	与 Dataflow Definition 数据流定义和 Metadataflow Definition 元数据流定义集的关联。
	+map	与 StructureMap 结构图的关联。
	+itemSchemeMap	与 ItemScheme Map 项目方案映射的关联。
StructureMap	衍生自 NameableArtefact	连接源和目标结构,在此源和目标结构是语义上的对等关系。
	sourceStructure	与源 Structure 结构的关联。
	targetStructure	与目标 Structure 结构的关联,目标 Structure 结构必须与源 Structure 结构同类型。
	sourceStructureUsage	与源 Structure Usage 结构用法的关联。
	targetStructureUsage	与目标 Structure Usage 结构用法的关联,目标 Structure Usage 结构用的必须与源 Structure Usage 结构用法同类型。

11.3 结构映射

11.3.1 类图

结构映射的类图见图32。

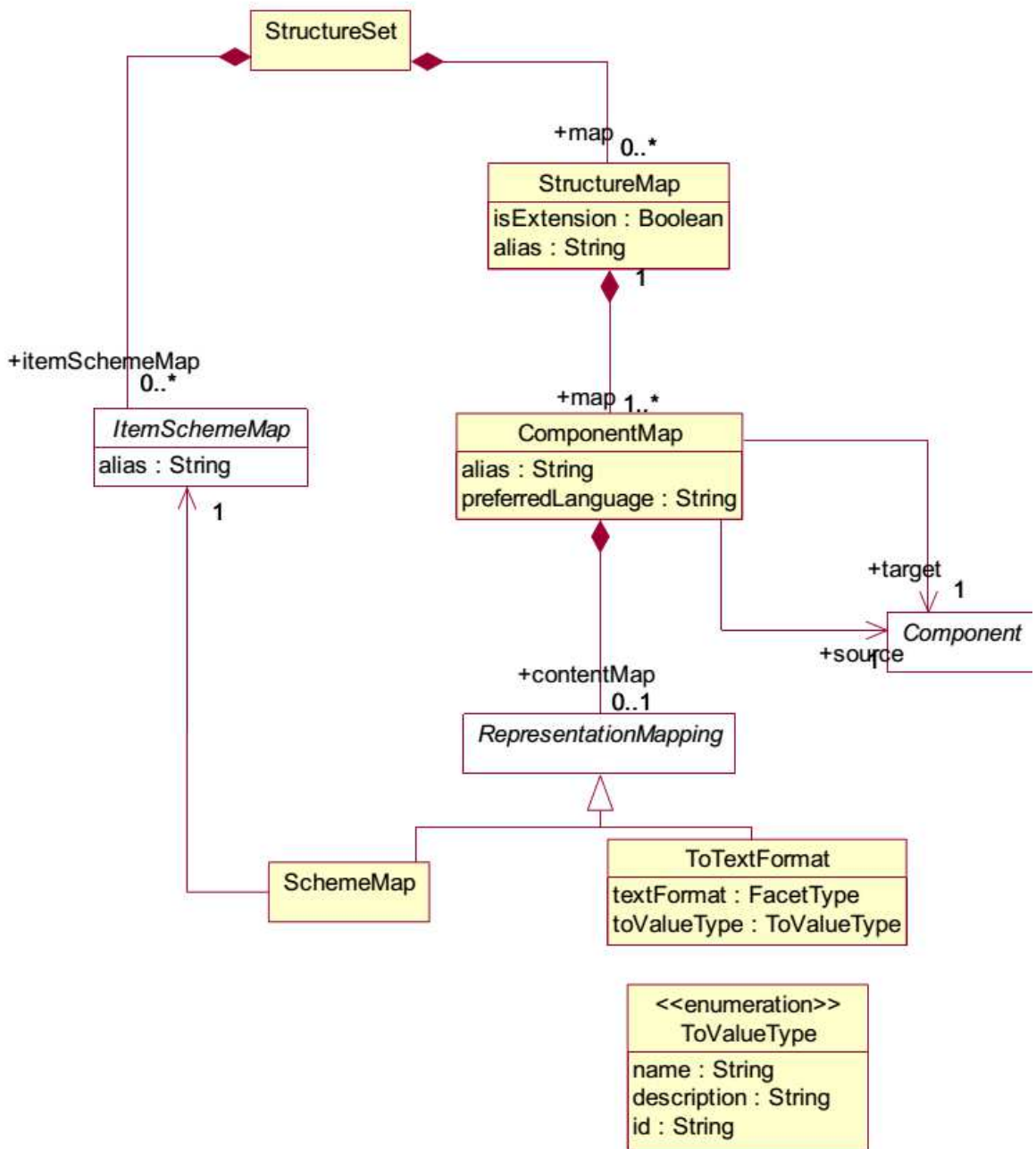


图32 结构映射的类图

11.3.2 图解

11.3.2.1 叙述

StructureMap 引用两个 *Structure* 或 *StructureUsage*。具体而言，这些引用应是 *DataStructureDefinitions*、*MetadataStructureDefinitions*、*DataflowDefinitions* 或 *MetadataflowDefinitions*。StructureMap 包含一个 ComponentMap 集，每一个指示出引用的 *Structure* 或 *StructureUsage* 的组件是同等的。ComponentMap 拥有属性 toTextFormat，其取值：id、name、description。这个指示映射工具使用身份标识、名字或一个代码组件的说明来

确定等值的非代码组件的值。对每一个指示的 *Component* 等值（这是有效的 *Concept* 等值），可能定义一个 *CodeListMap*。

ComponentMap 的例子是连接源 *Component*（这是源 DSD（*StructureMap* 里标识的）里的 *Dimension*）到相等的目标 *Component*（这是目标 DSD 里的 *Dimension*）。

11.3.2.2 定义

定义见表22。

表22 定义

类	特性	描述
<i>StructureMap</i>	衍生自 <i>NameableArtefact</i>	连接源和目标结构，在此源和目标结构是语义上的对等关系。
	<i>alias</i>	映射的交替认证，允许多重映射关系由该值的部分表示。
	<i>+map</i>	与 <i>Component Map</i> 组件映射的关联。
<i>ComponentMap</i>	衍生自 <i>AnnotableArtefact</i>	连接源和目标 <i>Component</i> 组件，在此源和目标 <i>Component</i> 组件是语义上的对等关系。
	<i>alias</i>	映射的交替认证，允许多重映射关系由该值的部分表示。
	<i>preferredLanguage</i>	指定到 <i>RepresentationMap</i> 表示映射的 <i>To Text Format</i> 文本格式选项内容所使用的语言。
	<i>+source</i>	与源 <i>Component</i> 组件的关联。
	<i>+target</i>	与目标 <i>Component</i> 组件的关联。
	<i>+contentMap</i>	与映射 <i>Components</i> 组件内容的构想关联-可以是 <i>Item Scheme</i> 项目方案的一个子级或到文本的映射。
<i>Representation Mapping</i>	抽象级 子级： <i>SchemeMap</i> <i>ToTextFormat</i>	定义源 <i>Component</i> 组件内容与目标 <i>Component</i> 组件内容间的映射。
<i>SchemeMap</i>	衍生自 <i>RepresentationMapping</i>	关联 <i>ItemScheme Map</i> 项目方案映射。
<i>ToTextFormat</i>	衍生自 <i>RepresentationMapping</i>	定义文本格式。
	<i>textFormat</i>	文本格式类型。
	<i>toValueType</i>	当将源 <i>Component</i> 组件内容映射到目标 <i>Component</i> 组件内容时，识别来自源 <i>Component</i> 组件的项目运用的构架。

ToValueType	在 Item 项目中构架的计数。
-------------	------------------

11.4 项目方案映射

11.4.1 背景

ItemSchemeMap项目方案映射用于关联在两个不同ItemSchemes项目方案中的Items项目。这是用于映射Items项目的通用原理。在ItemScheme项目方案中Items项目之间存在语义上的对等，对于映射这类项目有特定的模型。模型支持同类型的任何两个ItemSchemes项目方案的映射。它们是：

- ConceptScheme概念方案
- CategoryScheme类别方案
- OrganisationScheme组织方案
- Codelist代码表
- ReportingTaxonomy报告分类

11.4.2 类图

概念方案映射和类别方案映射的类图见图33。

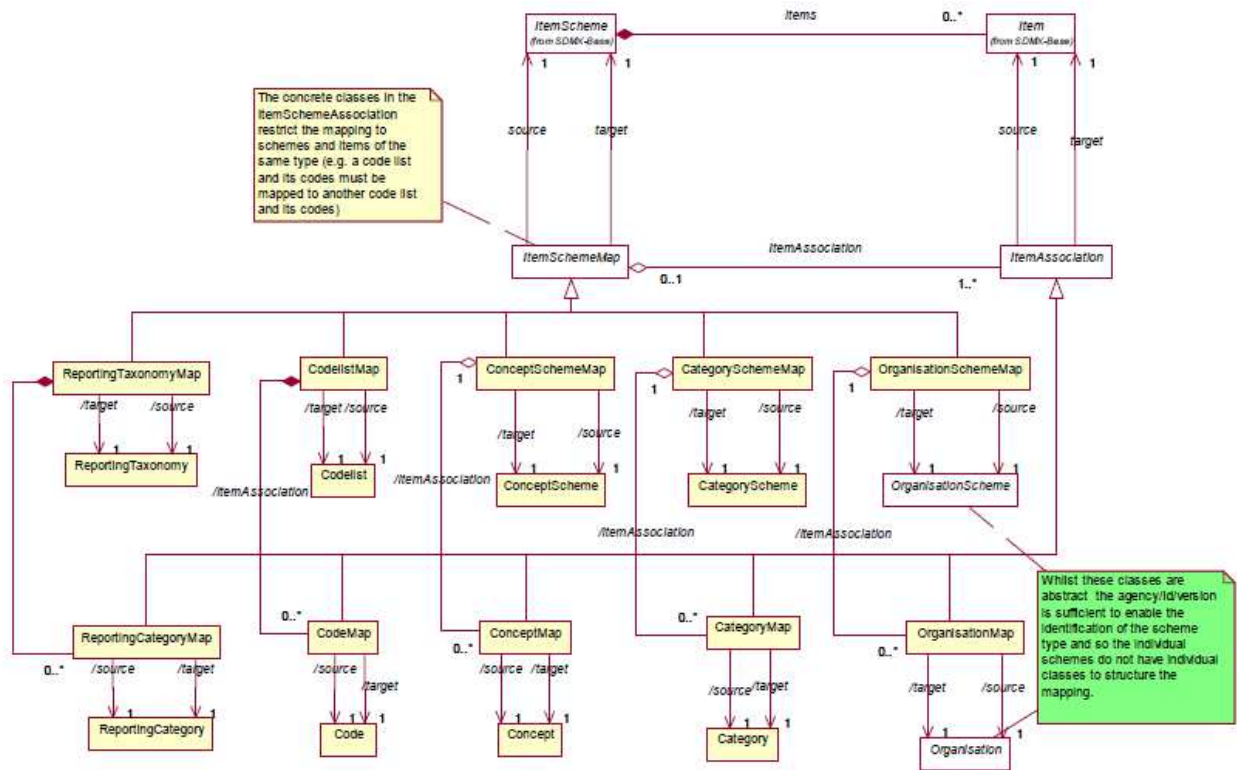


图33 项目方案映射类图

11.4.3 图解

11.4.3.1 叙述

ItemSchemeMap项目方案映射和ItemAssociation项目关联均衍生自NameableArtefact可命名产成品。

ConceptSchemeMap 概念方案映射、CategorySchemeMap 类别方案映射、OrganisationSchemeMap组织方案映射、CodelistMap代码表映射、ReportingTaxonomyMap

报告分类映射的每个均提供在列表中为指定项目 (Concept, Category, Code, Organisation, ReportingCategory) 间语义对等的机制。注意 OrganisationScheme 组织列表和 Organisation 组织的任何类型都可映射 (例如在 AgencyScheme 代理类别中的 Agency 代理可映射到 OrganisationUnitScheme 机构单位列表中的 OrganisationUnit 机构单位)。

每个列表映射识别一个内容被映射的 +source 源和 +target 目标列表。注意许多方案可通过配对方面映射的集组合到一起。ConceptMap 概念映射、CategoryMap 类别映射、CodelistMap 代码表映射、OrganisationMap 组织映射、ReportingTaxonomyMap 报告分类映射指示哪些 Concepts 概念, Categorys 类别, Codes 代码, Organisations 组织, 和 ReportingCategorys 报表分类是语义对等的, 同时可指定一个共享的别名以参照映射概念集, 方便查询。

11.4.3.2 定义

定义见表23。

表23 定义

类	特性	描述
<i>ItemSchemeMap</i>	衍生自 <i>NameableArtefact</i> 子级 ConceptSchemeMap CategorySchemeMap CodelistMap OrganisationSchemeMap ReportingTaxonomySchemeMap	关联两个 ItemSchemes 项目方案。
	source	与源 ItemSchemes 项目方案的关联。
	target	与目标 ItemSchemes 项目方案的关联。
	ItemAssociation	与 ItemAssociation 项目关联性的关联。
<i>ItemAssociation</i>	衍生自 <i>AnnotableArtefact</i> 子级 ConceptMap CategoryMap CodeMap OrganisationMap ReportingCategoryMap	
	source	与源 Item 项目的关联。
	target	与目标 Item 项目的关联。
ConceptSchemeMap	衍生自 <i>ItemSchemeMap</i>	关联源和目标 Concept Scheme 概念方案。

	/source	与源 Concept Scheme 概念方案的关联。
	/target	与目标 Concept Scheme 概念方案的关联。
ConceptMap	衍生自 <i>ItemAssociation</i>	关联源和目标 Concept 概念。
	/source	与源 Concept 概念的关联。
	/target	与目标 Concept 概念的关联。
CodelistMap	衍生自 <i>ItemSchemeMap</i>	关联源和目标 Code list 代码表。
	/source	与源 Code list 代码表的关联。
	/target	与目标 Code list 代码表的关联。
CodeMap	衍生自 <i>ItemAssociation</i>	关联源和目标 Code 代码。
	/source	与源 Code 代码的关联。
	/target	与目标 Code 代码的关联。
CategorySchemeMap	衍生自 <i>ItemSchemeMap</i>	关联源和目标 Category Scheme 分组方案。
	/source	与源 Category Scheme 分组方案的关联。
	/target	与目标 Category Scheme 分组方案的关联。
CategoryMap	衍生自 <i>ItemAssociation</i>	关联源和目标 Category 分组。
	/source	与源 Category 分组的关联。
	/target	与目标 Category 分组的关联。
OrganisationSchemeMap	衍生自 <i>ItemSchemeMap</i>	关联源和目标 OrganisationScheme 组织方案。
	/source	与源 OrganisationScheme 组织方案的关联。
	/target	与目标 OrganisationScheme 组织方案的关联。
OrganisationMap	衍生自 <i>ItemAssociation</i>	关联源和目标 Organisation 组织。
	/source	与源 Organisation 组织的关联。
	/target	与目标 Organisation 组织的关联。
ReportingTaxonomyMap	衍生自 <i>ItemSchemeMap</i>	关联源和目标 ReportingTaxonomy 报告分类。
	/source	与源 ReportingTaxonomy 报告分类的关联。

	/target	与目标 ReportingTaxonomy 报告分类的关联。
ReportingCategoryMap	衍生自 ItemAssociation	关联源和目标 ReportingCategory 报告分类。
	/source	与源 ReportingCategory 报告分类的关联。
	/target	与目标 ReportingCategory 报告分类的关联。

11.5 混合代码表映射

11.5.1 类图

见图34。

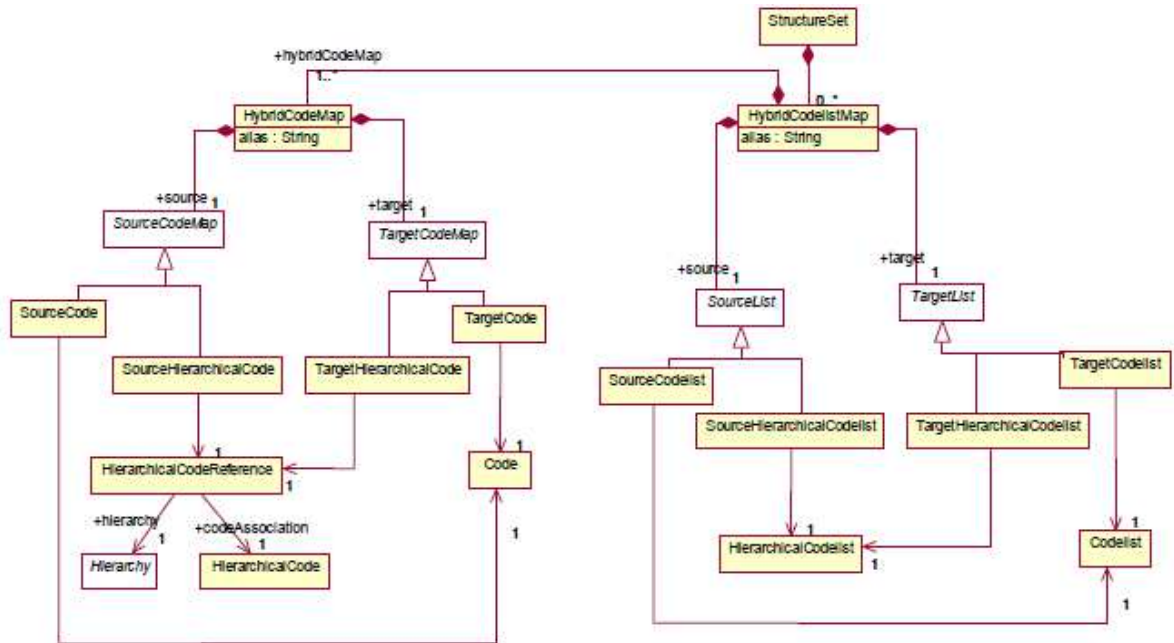


图34 混合 CodeList 映射类图

11.5.2 图表解释

11.5.2.1 叙述

HybridCodeListMap混合代码表映射绘制了CodeList代码表和HierarchicalCodeList层级代码表的内容。它包括在两个列表（HybridCodeMap混合代码映射）中的代码的映射。HybridCodeMap混合代码映射绘制了一个Code代码或一个HierarchicalCode层级代码与一个Code代码或HierarchicalCode层级代码的映射。HierarchicalCode层级代码依据Hierarchy层级和HierarchicalCode层级代码的结合来认证。

11.5.2.2 定义

见表24。

表24 定义

类	特性	描述
HybridCodelist Map	衍生自 <i>NameableArtefact</i>	关联 Codelist 代码表与 Hierarchical Codelist 层级代码表。
	alias	映射的交替认证, 允许多重映射关系由该值的部分表示。
	+source	与源 List 列表的关联。
	+target	与目标 List 列表的关联。
	+hybridCodeMap	在 Hybrid Codelist Map 混合代码表映射中与 Hybrid Code Maps 混合代码映射集的关联。
SourceList	抽象级 子级: SourceCodelist SourceHierarchical Codelist	
TargetList	抽象级 子级 TargetCodelist TargetHierarchical Codelist	
SourceCodelist		识别 Codelist 代码表, 这是映射的源。
TargetCodelist		识别 Codelist 代码表, 这是映射的目标。
SourceHierarchical Codelist		识别 HierarchicalCodelist 层级代码表, 这是映射的源。
TargetHierarchical Codelist		识别 HierarchicalCodelist 层级代码表, 这是映射的目标。
HybridCodeMap	衍生自 <i>AnnotableArtefact</i>	在 Hybrid Codelist Map 混合代码表映射中关联源和目标代码。
	+source	关联 SourceCode Map 源代码映射。
	+target	关联 TargetCode Map 目标代码映射。
SourceCodeMap	抽象级 子级 SourceCode SourceHierarchical Code	

TargetCodeMap	抽象级 子级 TargetCode TargetHierarchical Code	
SourceCode		识别 Code 代码, 这是映射的源。
TargetCode		识别 Code 代码, 这是映射的目标。
SourceHierarchical Code		识别 HierarchicalCode 层级代 码, 这是映射的源。
TargetHierarchical Code		识别 HierarchicalCode 层级代 码, 这是映射的目标。
HierarchicalCode Reference		在 Hierarchical Codelist 层级代 码表中, 参考 Hierarchy 层级和 Hierarchical Code 层级代码。
	+hierarchy +codeAssociation	在层级代码表的层级中, 关联层级 代码。

12 约束条件

12.1 范围

本部分内容是描述如何支持在元模型中规范数据源读取及其内容。这些信息一般储存在诸如应用程序使用的注册表之类的资源中, 以便定位通过互联网可获取的数据和元数据。Constraint也可用于指定在附加同Constraint的模块(例如, Data Structure Definition, Dataflow, Provision Agreement)环境相关的部分代码表的 Codelist子集。

注意, 在这个元模型中的数据源是指数据源和元数据源, 数据供应方是指数据供应方和元数据供应方。

数据源可以是包含数据或元数据的一份简单文件(使用SDMX-ML格式), 也可以是数据库或元数据库。数据源包括多个数据或元数据流的数据(在模型中叫做DataflowDefinition和MetadataflowDefinition), 本节中描述的机制允许组织精确地规定注册数据源内容的范围(SimpleDataSource和QueryDataSource)。

DataflowDefinition和MetadataflowDefinition本身可以被指定为仅包含由DataStructureDefinition或MetadataStructureDefinition派生的全部可能关键字的子集。

这些规范在本模型中叫做Constraint。

12.2 继承

12.2.1 可限模块类图-继承

可限和供应模块的继承类图见图35。

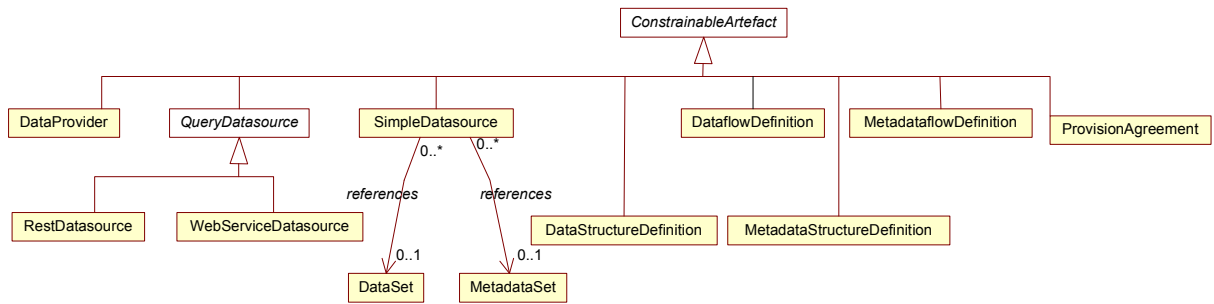


图35 可限和供应模块的继承类图

12.2.2 图解

由 *ConstrainableArtefact* 派生的任一模块均有定义的约束条件。可附加约束元数据的模块包括:

- *DataflowDefinition*
- *ProvisionAgreement*
- *DataProvider* (受限于发布日程)
- *MetadataflowDefinition*
- *DataStructureDefinition*
- *MetadataStructureDefinition*
- *DataSet*
- *SimpleDataSource* (注册时引用了实际 *DataSet* 或 *MetadataSet* 的注册数据源)
- *QueryDataSource*

注: 由于 *Constraint* 可以指定含有特定 *Structure* 的组件值的子集(例如, 特定的 *DataStructureDefinition* 或特定的 *MetadataStructureDefinition*), 因此 *ConstrainableArtefacts* 必须具有特定 *Structure*。因此, 即便 *Constraint* 本身可能不直接与 *DataStructureDefinition* 或 *MetadataStructureDefinition* 关联, 但其约束的模块会与 *DataStructureDefinition* 或 *MetadataStructureDefinition* 关联。因为 *Data Provider* 不与任何一个特定 *DSD* 或 *MSD* 关联, 关联至 *Data Provider* 的 *Constraint* 包含信息的类型受限于 *Release Calendar*。

12.3 约束条件

12.3.1 关系类图-高级视角

约束元数据的关系类图见图36。

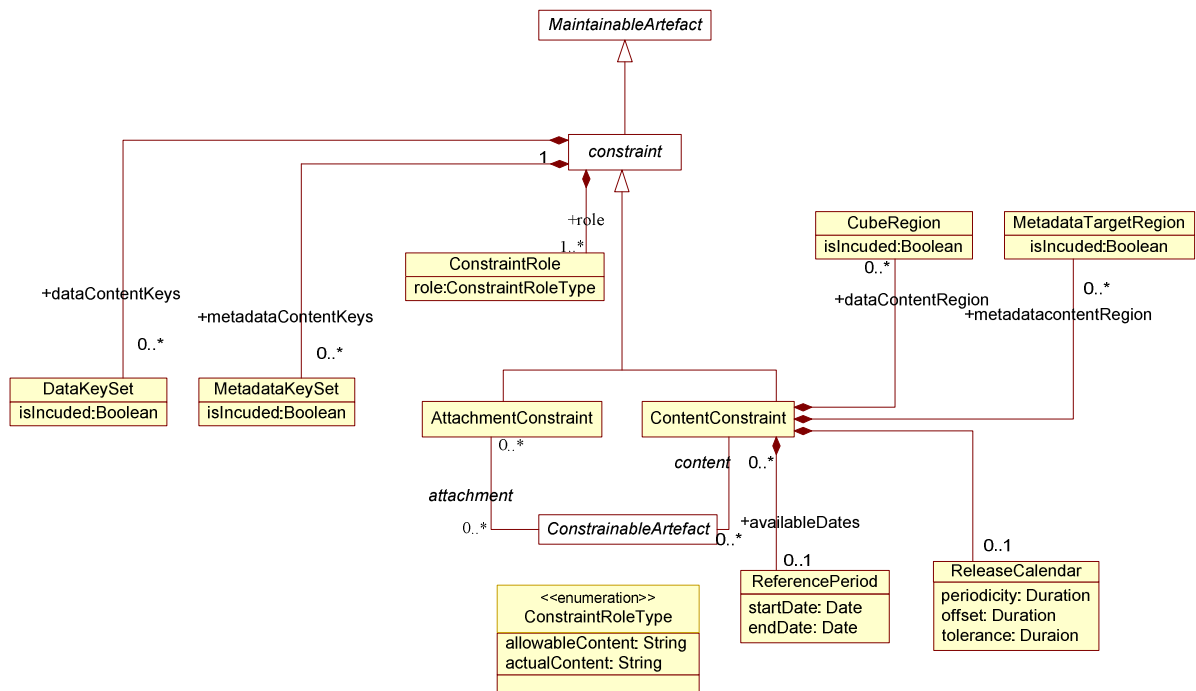


图36 约束元数据的关系类图

12.3.2 图解

约束机制允许为 *ConstrainableArtefact* 附加特定约束条件。除了 *ReferencePeriod* 和 *ConstrainableArtefact*，这些约束条件指定了可在任何 *ConstrainableArtefacts* 中出现的值或关键字全集的子集。

例如，*DataStructureDefinition* 从每个 *Dimension* 上指定了允许代码值列表。然而，使用 *DataStructureDefinition* 的特定 *DataflowDefinition* 仅包含可能关键字范围的子集，理论上这些关键字来自 *DataStructureDefinition* 定义（全部可能性有时被称作维度值的 Cartesian 乘积）。此外，能够依照 *DataflowDefinition* 提供数据的数据提供者拥有 *ProvisionAgreement*。为描述数据供应方能够提供的数据，*DataProvider* 可能同时希望提供可以进一步约束可能性范围的信息。从其包含的 *KeySets* 或 *CubeRegions* 方面对数据源内容进行描述也是有用的。

ConstrainableArtefact 有两类 *Constraint*：

ContentConstraint：仅作为一种机制使用，用于指定可用关键字集（*DataKeySet*、*MetadataKeySet*）、诸如 *DataSet* 或数据库（*QueryDataSource*）的 *DataSource* 中的组件值集（*CubeRegion*、*MetadataTargetRegion*）或者是从 *DataStructureDefinition* 得到的允许关键字。*ConstrainableArtefact* 中存在多种约束条件。例如，一项 *ContentConstraint* 指定了 *ConstrainableArtefact* 的允许赋值（*role* 为 *allowableContent*），其可用于验证或创建局部代码表。同时，另一项约束条件可以指定数据或元数据源的实际内容（*role* 为 *actualContent*）。

AttachmentConstraint：作为一种机制，用于定义数据全集中的片段，在 *DataSet* 或 *MetadataSet* 中元数据附加在这些片段上。这些片段既可以定义为关键字集（*KeySet*），也可以定义为组件值集（*CubeRegion*）。对特定的 *AttachableArtefact*，可以指定多个 *AttachmentConstraint*。

除了 *DataKeySet*、*MetadataKeySet*、*CubeRegion*、*MetadataTargetRegion* 之外，约束条件还包括 *ReferencePeriod*，其定义一个或多个数据范围（*ValidityPeriod*），明确了在

*ConstrainableArtefact*中可以获得数据或元数据的时期，以及指定了数据对外公布和报告时间的ReleaseCalendar

12.3.3 关系类图-详细

关键字集约束条件见图37，空间区域和元数据目标区域约束条件见图38。

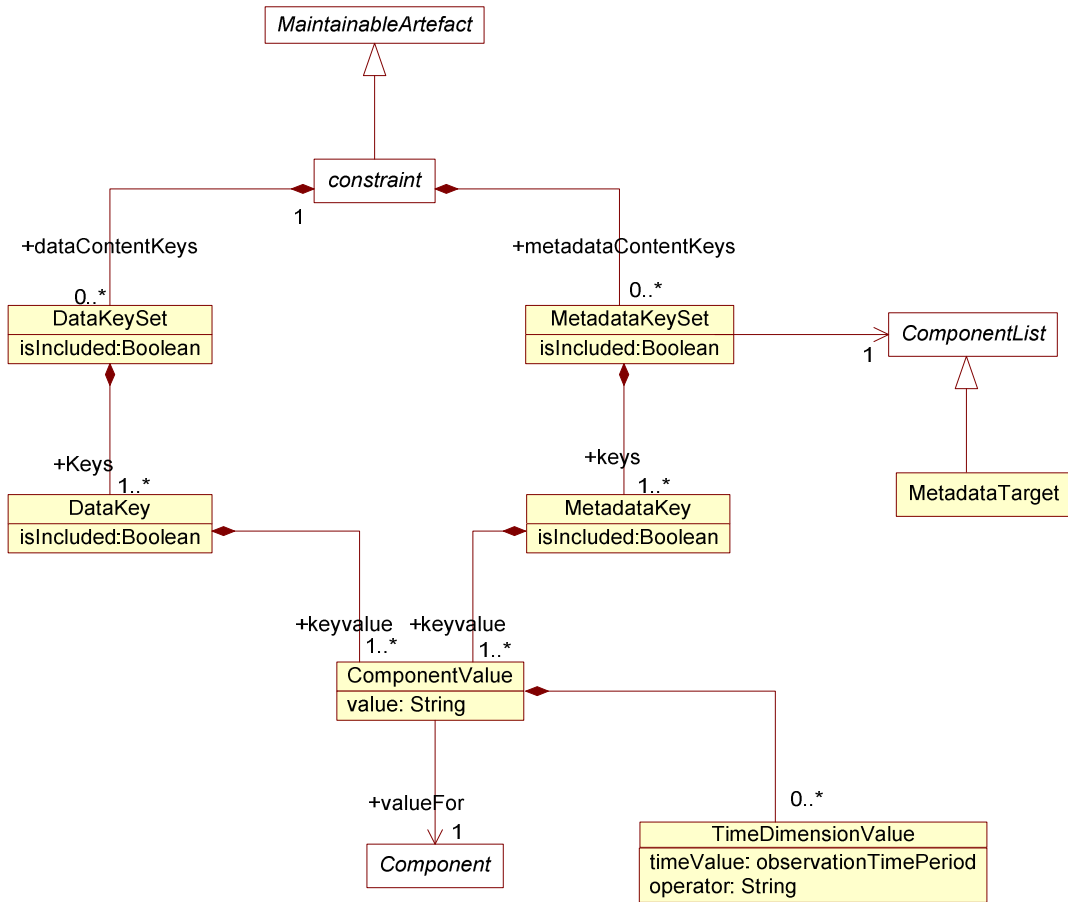


图37 约束条件-关键字集约束条件

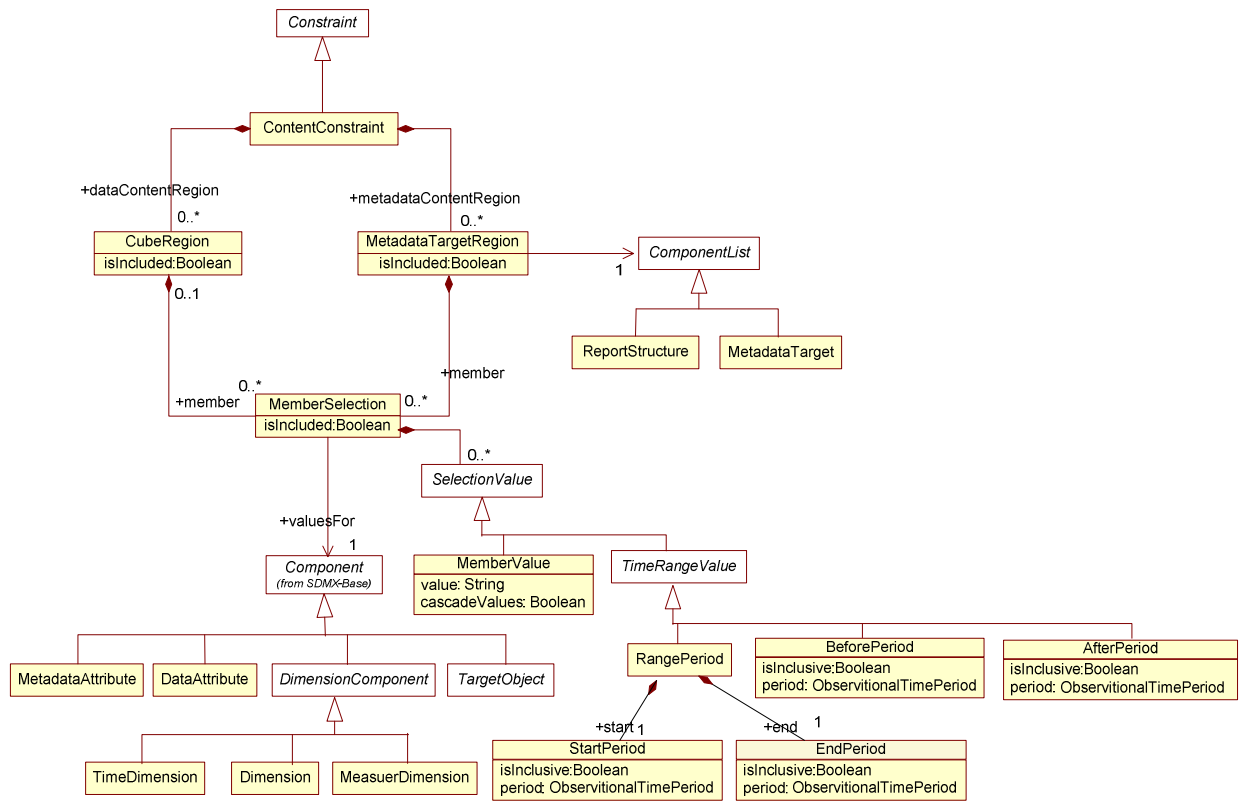


图38 约束条件-空间区域和元数据目标区域约束条件

12.3.3.1 图解

Constraint是一个MaintainableArtefact。

对于Constraint，有两种指定值子集的方式可选：

- a) 作为在DataSet (DataKeySet) 或MetadataSet (MetadataKeySet) 中出现的关键字集。每一个 DataKey 或 MetadataKey 指定了多个 ComponentValues，每一个 ComponentValues都引自Component (例如，Dimension、TargetObject)。当包含在DataSet或MetadataSet中时，每个ComponentValue可以作为某一结构的Component表示值。由于 MetadataStructureDefinition 中有许多 MetadataTarget，MetadataKeySet 必须能够对其进行识别。对于DataKeySet来说，因为只有一个 DimensionDescriptor和 AttributeDescriptor，没有必要进行等价识别。
- b) 作为CubeRegions或MetadataTargetRegions集，每个集都以一个或多个MemberValues的形式定义整体结构 (MemberSelection) 的“片段”，当包含于DataSet或MetadataSet中时，这些MemberValues可表示一个结构的Component。

以上a) 和b) 的区别在于：a) 中定义了全部关键字，而b) 中“片段”为每一个Component定义了可能值列表，但是没有指定特定的关键字组合。此外，a) 中Component同DataKeyValue或MetadataKeyValue之间的关联受限于由关键字和标识符构成的组件，然而在b) 中关联包含其他组件类型 (例如属性)。ComponentValue.value 和 MemberValue.value 值必须与在DataStructureDefinition 和 MetadataStructureDefinition 中Component声明的Representation一致。注意，在任何情况下，Value上的运算符视为“等号”。此外，在MemberValue中可以通过cascadeValues属性来明确子值 (例如，子代码) 包含在约束条件中。

不论约束条件定义中包含 (isIncluded="true") 还是不包含 (isIncluded="false") DataKeySet、DataKey、MetadataKeySet、MetadataKey、CubeRegion、MetadataTargetRegion和MemberSelection, 都可以对这些集进行定义。这种属性是有用的, 例如, 只是可能值的一个小子集没有包含在集中, 那么这个小子集可以被定义, 并且排除在约束条件之外。注意, 如果子模型被“包括”在内, 父模型被“排除”在外, 那么, 子模型包含在被“排除”的模型列表之中。

12.3.3.2 定义

定义见表25。

表25 定义

类	特征	说明
Constrainable Artefact	抽象类 子类是: DataflowDefinition Metadataflow Definition ProvisionAgreement DataProvider QueryDatasource SimpleDatasource DataStructure Definition MetadataStructure Definition	有特定约束条件的模块。
	content	与元数据关联, 该元数据约束链接至Constrainable Artefact的数据或元数据源中的内容。
	attachment	与元数据关联, 该元数据约束可以附加元数据的Constrainable Artefact的有效内容。
Constraint	衍生自 MaintainableArtefact 抽象类. 子类是: AttachmentConstraint ContentConstraint	指定了数据或元数据源允许或实际内容定义的子集, 该数据源派生自定义了代码表和其他有效内容的Structure。
	+availableDates	关联至时期, 该时期识别在数据源中可以获取数据或元数据的时间范围。
	+dataContentKeys	关联至Data Key Sets (即值组合)的子集。Data Key Sets派生自与Constrainable Artefact链接的结构定义。

	+metadataContentKeys	关联至Metadata Key Sets (即值组合) 的子集。Data Key Sets派生自与Constrainable Artefact链接的结构定义。
	+dataContentRegion	关联至组件值子集。组件值派生自与Constrainable Artefact链接的Data Structure Definition。
	+metadataContentRegion	关联至组件值子集。组件值派生自与Constrainable Artefact链接的Metadata Structure Definition。
ContentConstraint	衍生自 Constraint	使用内容定义Constraint, 这些内容可以在与Constrainable Artefact链接的数据或元数据源中找到, 且这个约束条件与Constrainable Artefact关联。
	+role	关联至Constraint扮演的角色。
ConstraintRole		按照Constraint的目的指定其内容的类型。
	allowableContent	Constraint包含Component值或关键字有效子集规范。
	actualContent	Constraint包含数据或元数据源的实际内容规范, 并用数据源中的Component值或关键字表示。
Attachment Constraint	衍生自 Constraint	使用在数据源中的组件值组合定义Constraint, 且在结构定义中Constrainable Artefact与Constraint相关。
DataKeySet		数据关键字集。
	isIncluded	表示Data Key Set是包含在约束条件定义之中还是不包含在约束条件之中。
	+keys	关联至集中的Data Keys。
MetadataKeySet		元数据关键字集。
	isIncluded	表示Metadata Key Set是包含在约束条件定义之中还是不包含在约束条件之中。
	+keys	关联至集中的Metadata Keys。
DataKey		数据集中的关键字值。
	isIncluded	表示Data Key 是包含在约束条件定义之中还是不包含在约束条件之中。

	+keyValue	与由关键字构成的Component Values关联。
MetadataKey		元数据集中的关键字值。
	isIncluded	表示Metadata Key 是包含在约束条件定义之中还是不包含在约束条件之中。
	+keyValue	与包含关键字的Component Values关联。
ComponentValue		关键字的Component标识和值（例如，Dimension）。
	value	Component值。
	+valueFor	关联至与Constrainable Artefact链接的Structure中的Component（例如，Dimension）。
TimeDimensionValue		Time Dimension组件值。
	timeValue	时期值。
	operator	表示特定值是否代表确切时间还是时期，或者该值是否应该作为一个时间区间处理。 greaterThan或greaterThanOrEqual值分别表示不包括或包括时间区间的起始时点。 lessThan或lessThanOrEqual值分别表示不包括或包括时间区间的结束时点。 当时间区间指定边界没有对边界时，该对边界当作无穷处理（例如，对于greaterThanOrEqual来说，表示给定时间区间之后的任何时期）。
CubeRegion		Component集及Component值集，其定义了与Constrainable Artefact链接的数据结构的全部可能内容的子集或“片段”。
	isIncluded	表示Cube Region是包含在约束条件定义之中还是不包含在约束条件之中。
	+member	与定义了值子集的Component集关联。
MetadataTargetRegion		Component集及Component值集，其

		定义了与Constrainable Artefact链接的元数据结构的全部可能内容的子集或“片段”。
	isIncluded	表示Metadata Target Region是包含在约束条件定义之中还是不包含在约束条件之中。
	+member	与定义了值子集的Component集关联。
MemberSelection		轴上组件的允许值集。
	isIncluded	表示Member Selection是包含在约束条件定义之中还是不包含在约束条件之中。
	+valuesFor	关联至与Constrainable Artefact链接的Structure Component, 其为Member Values 定义了有效的Representation.
SelectionValue	抽象类.子类: MemberValue TimeRangeValue	Member Selection的一组值, 与其他Member Selections构成Cube Region的内容值。
MemberValue	衍生自 SelectionValue	Member Selection值集合中的某一个单独值。
	value	成员值。
	cascadeValues	表示成员子节点包含在Member Selection中(例如, 子代码)。
TimeRangeValue	衍生自 SelectionValue 抽象类 具体类 BeforePeriod AfterPeriod RangePeriod	指定了日期或多个日期的时间值, 带约束条件的筛选对于该日期或多个日期是有效的。
BeforePeriod	衍生自 TimeRangeValue	时期, 在该时期前带约束条件的筛选有效。
	isInclusive	表示日期是否包括在时期之内。
AfterPeriod	衍生自 TimeRangeValue	时期, 在该时期后带约束条件的筛选有效。
	isInclusive	表示日期是否包括在时期之内。
RangePeriod		时间区间的起止时期。
	+start	关联至Start Period。
	+end	关联至End Period。
StartPeriod	衍生自 TimeRangeValue	时期, 自此时期起带约束条件的筛选有效。

	isInclusive	表示该日期是否包括在时期之内。
EndPeriod	衍生自 TimeRangeValue	时期,至此时期止带约束条件的筛选有效。
	isInclusive	表示该日期是否包括在时期之内。
ReferencePeriod		时间集,其约束数据或元数据集中的内容。
	startDate	时期起始日期。
	endDate	时期结束日期。
ReleaseCalendar		数据或元数据发布或报告的日程。
	periodicity	数据或元数据发布周期。
	offset	1月1日与数据首次发布日的间隔时间。
	tolerance	时期,过了该时期,数据或元数据可以视为延迟。

13 数据提供

13.1 类图

数据提供关系及继承类图见图39。

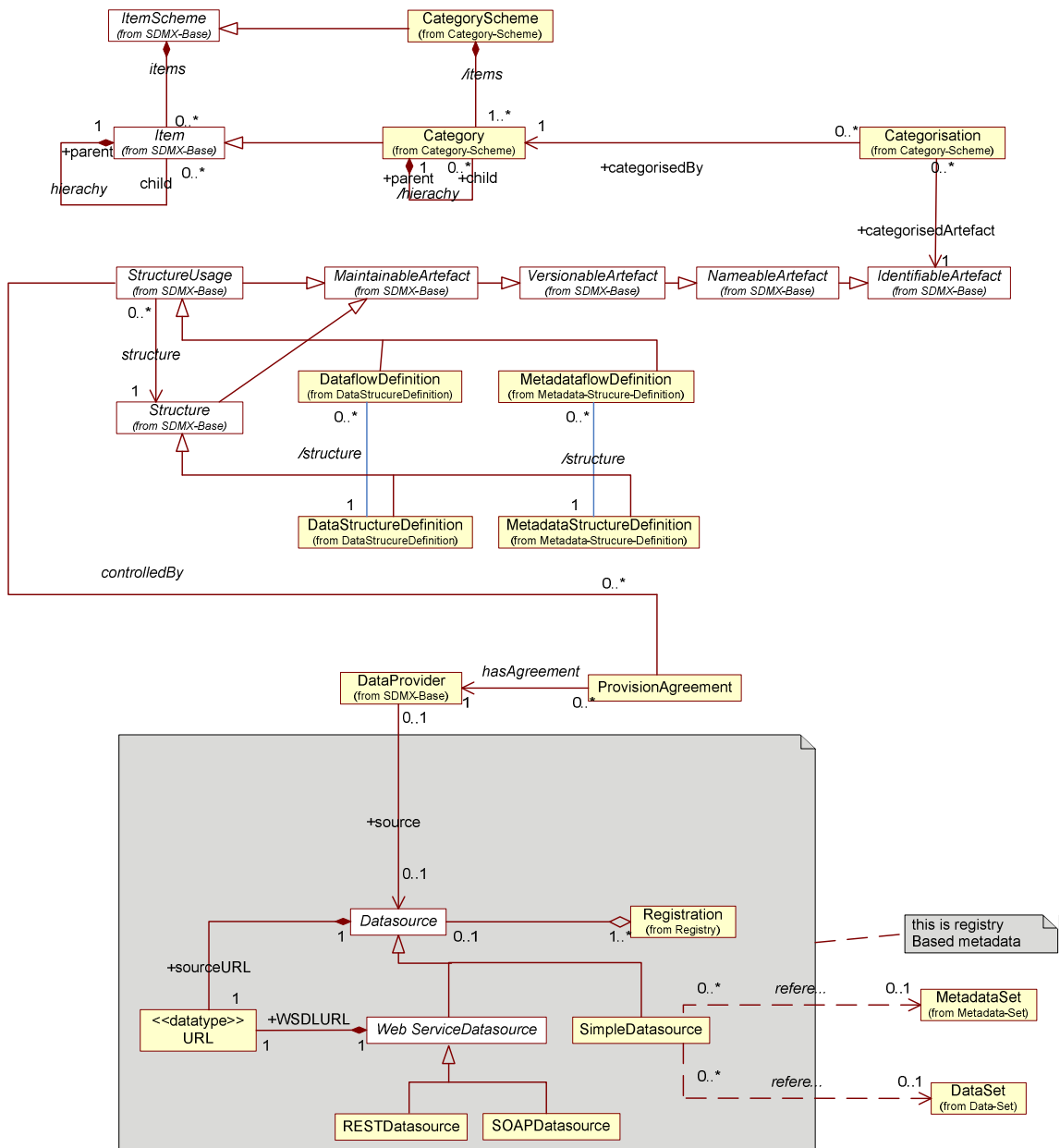


图39 数据提供关系及继承类图

13.2 图解

13.2.1 叙述

子模型将SDMX-IM中的多个模块连接在一起，由于子模型中的所有模块必须向数据、元数据注册应用程序或访问数据、元数据应用程序开放，因此子模型对SDMX元数据注册表至关重要。

同时，注册表包含了上图中的全部元数据，灰色阴影区域中的类被指定于供数据源（实际数据、元数据集或数据库、元数据库）注册的注册表场景之中。关于这些类是如何在注册表场景下使用的详情可以在SDMX 注册表接口标准中找到（SDMX标准的第五部分）。

ProvisionAgreement将定义了如何对数据和元数据进行结构化和分类（StructureUsage）的模块链接至DataProvider，并通过对数据或元数据注册的方式引用Datasource（可以是数据或

元数据），不论该数据源是网站上的符合SDMX的文件（SimpleDatasource）还是能够提供SDMX查询和反馈符合SDMX文件的数据库服务（QueryDatasource）。

拥有具体类 DataStructureDefinition 和 MetadataStructureDefinition 的 StructureUsage，与对应的DataflowDefinition和MetadataflowDefinition有关联，通过 Categorisation，可以将StructureUsage链接至CategoryScheme中的一个或多个Category（例如，作为StructureUsage分类依据的主题领域方案）。这样做可以对从主题领域中查找到相关的数据或元数据有所帮助。

SimpleDatasource链接至网站上的实际DataSet或MetadataSet(在图中显示为一个叫做“参考”的依赖关系)。在DataSet或MetadataSet的注册过程中，可获取sourceURL。更多关于SimpleDatasource内容的信息以Registration的ContentConstraint(见10.3)形式存储于注册表中。

QueryDatasource是抽象类，其代表了一种数据源，这种数据源可接受SDMX-ML查询（SOAPDatasource）或RESTful查询（RESTDatasource），并做出适当反馈。每一个不同的Datasource均从Datasource继承了dataURL，QueryDatasource拥有额外的URL，用于定位WSDL或WADL文件，以便描述如何获取QueryDatasource。假定所有其他支持协议都使用SimpleDatasource URL。

图40通过图形解析的方式展示了SDMX结构化模块间的重要传导关系，这些模块最终链接至数据或元数据的注册。

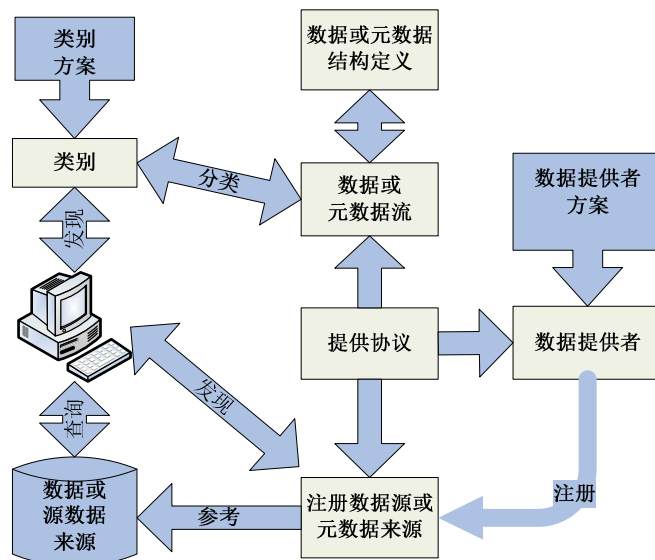


图40 结构化元数据同数据及元数据注册的链接图解

13.2.2 定义

定义见表26。

表26 定义

类	特征	说明
StructureUsage	抽象类： 子类是： DataflowDefinition MetadataflowDefinition	在Base中有相关描述。
	controlledBy	关联至Provision Agreement，

		该Provision Agreement包含与数据提供相关的元数据。
DataProvider		参见Organisation Scheme。
	hasAgreement	关联至Provision Agreement, 供应方为Provision Agreement提供数据或元数据。
	+source	关联至能够处理数据或元数据查询的数据或元数据源。
ProvisionAgreement		将Data Provider链接至相关的Structure Usage (例如: Dataflow Definition或Metadataflow Definition), 供应方为其提供数据或元数据。通过使用Constraint, 协议可以约束提供数据或元数据的范围。
	+source	关联至能够处理数据或元数据查询的数据源或参考元数据源。
Datasource	抽象类 子类是: SimpleDatasource WebServices Datasource	对可以获取数据或参考元数据的位置或服务进行的确认。
	+sourceURL	数据源和参考元数据源的URL (文件或Web服务)。
SimpleDatasource		SDMX-ML数据集, 其可以在URL中以文件形式获取。
WebServices Datasource	抽象类 衍生自 Datasource 子类是: RESTDatasource SOAPDatasource	能够处理数据或元数据查询的数据源或参考元数据源。
RESTDatasource		数据源和参考元数据源, 其可通过RESTful web服务接口获取。
SOAPDatasource		数据源和参考元数据源, 其同SOAP web服务接口保持一致。
	+WSDLURL	关联至Web Service Definition Language (SOAP) 或Web Service Application

		Language (REST) 网络服务配置文件的URL。
Registration		这里不详述该内容，但是将其作为SDMX-IM与Registry Service API之间的链接。它表示数据和元数据注册文件。

14 进程

14.1 介绍

在任何处理数据和引用元数据的系统中，系统本身就是一系列的进程，同时每个进程中，数据或参考元数据可经过一系列转换。这种进程尤其体现在原始数据到发布数据和参考元数据的转换过程中。在此提出的进程模型是通用模型，可以通过使用关联特定可识别对象及标识使用软件组成部分的方法，用文本或更加正式的方式，获取关于这些阶段的关键信息。

14.2 模型-继承和关系视角

14.2.1 类图

进程和转换的继承和关系类图见图41。

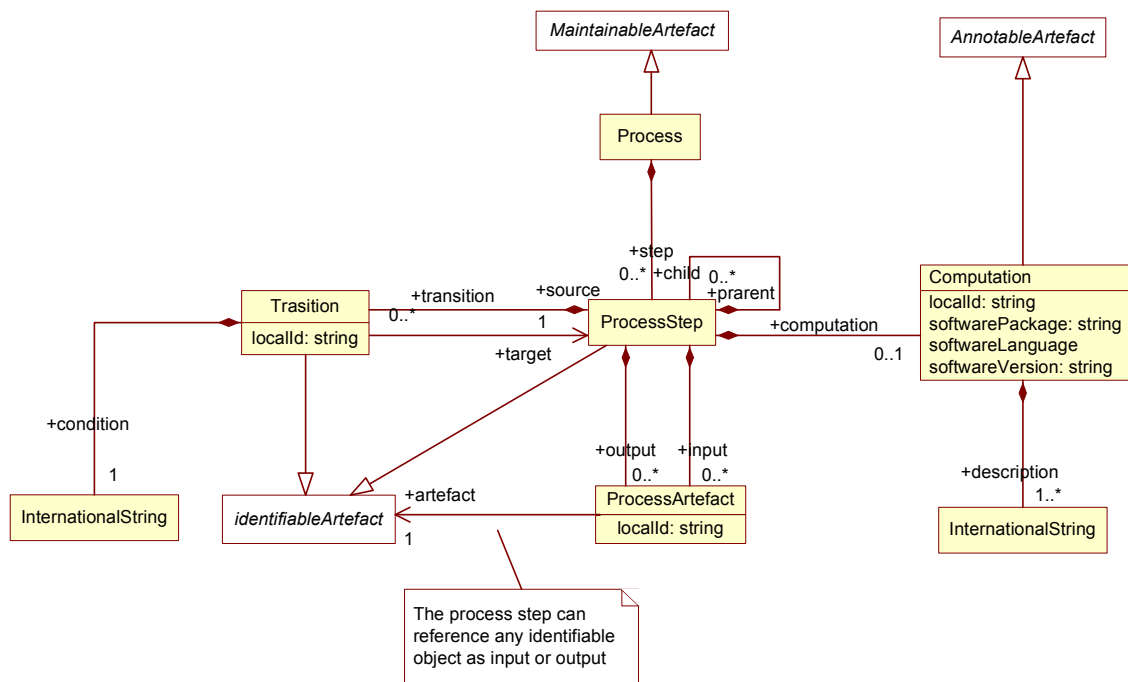


图41 进程和转换的继承和关系类图

14.2.2 图解

14.2.2.1 叙述

Process 是分层的 ProcessSteps 集。每个 ProcessStep 可以取 0 个或多个 *IdentifiableArtefacts* 为输入和输出对象。对每个对输入和输出的 *IdentifiableArtefacts* (ProcessArtefact) 关联可分配一个 localID。

由 ProcessStep 执行的计算可以选择性地使用 Computation 来描述，该计算可识别 ProcessStep 使用的软件，同时也能使用多种语言变量以文本形式 (+description) 进行描述。基于可用多语言变量描述的 +condition 结果，Transition 描述了从 +source ProcessStep 到 +target ProcessStep ProcessSteps 的执行。

14.2.2.2 定义

定义见表27。

表27 定义

类	特性	描述
Process	衍生自 <i>Maintainable</i>	一种方案,该方案定义或记录了对数据或元数据执行的操作,以便根据给定的规则验证数据或元数据,获取新的信息。
	+step	用来关联ProcessSteps。
ProcessStep	衍生自 <i>IdentifiableArtefact</i>	对数据或元数据执行的特定操作,以便根据给定的规则验证数据或元数据,获取新的信息。
	+input	关联至用来识别Process Step 输入对象的Process Artefact。
	+output	关联至用来识别Process Step 输出对象的Process Artefact。
	+child	关联至那些合并以形成本Process 一部分的子Processes。
	+computation	关联至一个或多个Computations。
	+transition	关联至一个或多个Transitions。
Computation		用文本形式描述进程中涉及的计算。
	localId	用以区别在同一Process中的 Computations。
	softwarePackage softwareLanguage softwareVersion	执行计算的软件信息。
	+description	用来对Computation进行描述或提供更多信息的文本。可使用多种语言变量。
Transition	衍生自 <i>IdentifiableArtefact</i>	使用文本或正式形式表述的两种特定数据操作之间的数据转换。

	+target	关联至作为Transition目标的ProcessStep。
	+condition	用来关联Transition的文本描述。
ProcessArtefact		对ProcessStep输入或输出对象的识别。
	+artefact	关联至作为Process Step的输入或输出对象的Identifiable Artefact。

15 转换和表达式

15.1 范围

模型中此部分的目的是能够追溯数据的派生。这与数据库中的世系概念类似——即数据是如何派生的。

模型中此部分的功能是允许对执行的运算（这些运算正常情况下是自动执行的，程序计算）进行识别和记录，并允许定义支持中性句法表达式“语法”的结构，这些语法可以从更细微的层次来详细说明运算，这样的话程序就可以“读取”元数据并使用任何合适的计算机语言生成需要的表达式。

模型此部分同时允许详细说明和记录不同数据间的一致性规则，并用运算式表示（例如，一致性规则：“ $a + b = c$ ”可写为“ $a + b - c = 0$ ”，并通过计算进行验证“if(($a + b - c$) = 0, then..., else ...)”。

需要注意的是下面的模型在范围和内容上与由Object Management Group (OMG)开发的Common Warehouse Metamodel (CWM)中的 Expression元数据模型类似。相关说明可在<http://www.omg.org/cwm>中找到。

CWM规范的第一部分的8.5章节中有对Expression 元数据模型的描述。下面的类图描述了SDMX-IM基础类中表述的CWM Expression metamodel。

15.2 模型-继承视角

15.2.1 类图

类转换的继承和关系类图见图42。

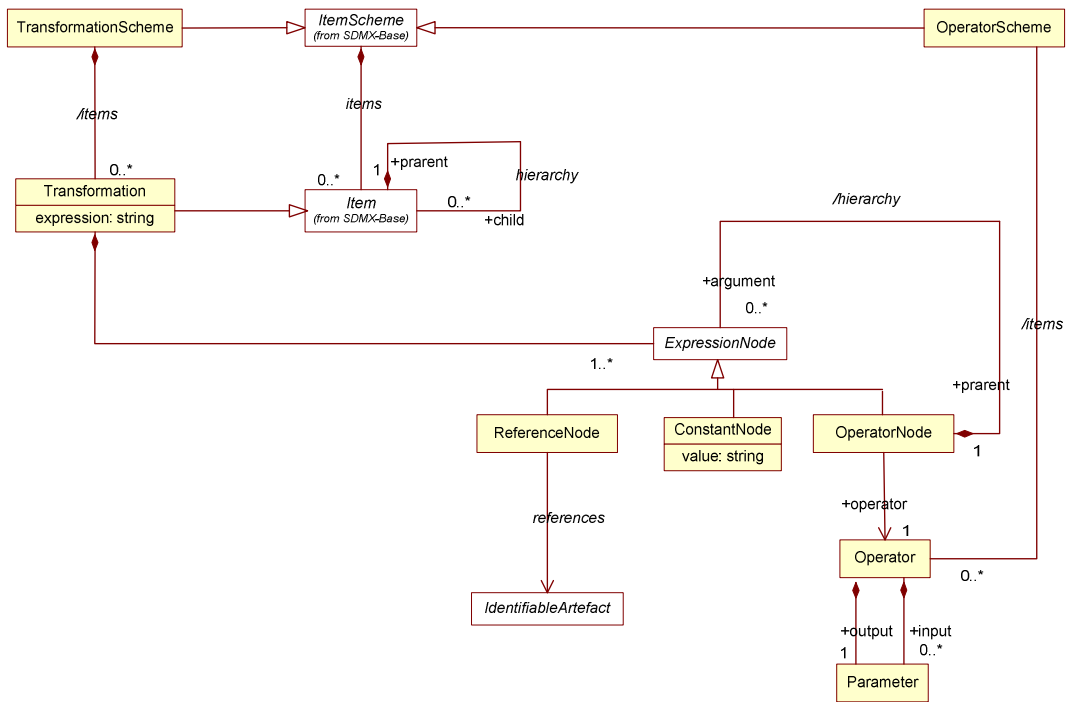


图42 类转换的继承和关系类图

15.2.2 图解

15.2.2.1 叙述

与本模型相关的 *ItemScheme* 有三种：

- a) *TransformationScheme*，由一个或多个 *Transformations* 构成。
- b) *OperatorScheme*，由一个或多个 *Operators* 构成。
- c) *ExpressionNodeScheme*，包含一个或多个 *ExpressionNodes*。

这里的模型是一个基础框架，可用于表达式和转换，但是需要做更多的工作以详细说明其是如何整合至模型中和在模型中实际应用。详细说明将在未来公布的标准中予以体现。

SDMX-IM 中的表达式概念采用了表达式树状图的功能视角，以达到使用相对少的节点类型来代表宽范围的表达式的效果。表达式层级图中出现的每一个函数或传统数学运算符在与 *Operator* 的关联中由 *+operator* 角色代表，并依次包含输入和输出参数。例如，算数加法运算“ $a + b$ ”可被认为是函数“ $\text{sum}(a, b)$ ”。“ sum ”是 *Operator*，“ a ”和“ b ”是它的 *Parameters*。参数是一项运算中通用的输入和输出对象的可能值（例如，底数和指数是幂运算的参数），然而在特定计算中，自变量是特定的参数值，（例如，在 Einstein 方程式“ $E = MC^2$ ”中，幂运算中的自变量是“ C ”（底数）和“ 2 ”（指数））。特定函数或运算的实际语义由使用的特定工具来决定，在 SDMX-IM 中不涉及。

SDMX-IM 中表达式描述的层次性由 *OperatorNode* 关联的递推性实现。该关联允许将一个表达式内的子层次视为他们父节点的实际自变量。

该模型能够用于定义数据派生和定义完整性检查（例如， A 与 B 的和必须等于 C ）。

尽管该模型定义用于包含中性语法表达式的数据结构，模型本身并不指定中性语法表达式的语法。可使用文本形式描述函数，这种描述可以使用无结构的函数解释，也可以使用像 BNF (Bankus Naur Form) 一样的更正式语言。

数据结构起如下作用：

需要执行的实际基础数学函数（如加，乘，除，等于(=)，<，>等）在OperatorScheme中定义为Operators。对于每个Operator，输入和输出Parameters在Parameter类中定义。

运算在TransformationScheme中定义为Transformations。Transformation是一个特定的运算，并通过使用表达式来详细说明，表达式则是通过按照所需顺序应用1个或者更多Operators生成（例如，使用文本格式，使用括号），并列明了Operators的Parameters实际自变量；整个表达式的结果等于作为Transformation结果的模型项目（如Einstein 方程式中的“E”）。Transformation对已有的IdentifiableArtefacts进行运算，运算结果是另一个IdentifiableArtefact。经运算的IdentifiableArtefacts此时可以是其他Transformations的运算对象。

Transformation表达式（例如，对于Einstein等式微积分，“E=M*(C**2)”）可以在ExpressionNodes层级中被分解（在例子中，“M”，“C”，“2”，“*”，“**”）。ExpressionNode可以是ReferenceNode，ConstantNode或OperatorNode。ReferenceNode引用identifiable model artefact（本例中，“M”和“C”）。按照定义，ConstantNode是常数值（本例中，“2”）。OperatorNode引用OperatorScheme中的Operator（本例中，“*”，“**”）。Transformation与其组成部分ExpressionNodes关联。

ExpressionNodes的层级关系告诉我们运算符在表达式中应用的顺序，是通过OperatorNode class的/hierarchy关联实现的，其中子ExpressionNodes是父OperatorNode的自变量。子ExpressionNodes必须以正确顺序与父OperatorNode引用的Operator正式参数相对应。（子ExpressionNode可以是另一个运算（指另一个OperatorNode）的结果，Constant或者IdentifiableArtefact（ReferenceNode）的参数。在SDMX-IM中，所有IdentifiableArtefacts有唯一的urn（Uniform Resource Name, 统一资源名称），urn由能够识别单独对象的值组成。这种urn的结构在注册表规范中定义。一个例子是代码的urn，它由agency:code-list-id.code-id组成—一个实例为
"urn:sdmx:org.sdmx.infomodel.codelist.Code=TFFS:CL_AREA(1.0).1A"。

15.2.2.2 定义

定义见表28。

表28 定义

类	特性	描述
Transformation Scheme	衍生自 <i>ItemScheme</i>	一种方案,该方案定义或记录所需转换,以便从其他数据中派生或验证数据。
Transformation	衍生自 <i>Item</i>	一个独立的Transformation。
	+expressionComponent	关联至Expression Node。
<i>ExpressionNode</i>	抽象类 子类 ReferenceNode ConstantNode OperatorNode.	共同定义或记录一个表达式的可能的分层节点中的节点。
	/hierarchy	关联至子Expression Node。

ReferenceNode	衍生自 <i>ExpressionNode</i>	引用特定对象的特定类型 Expression Node。
	references	关联至引用对象Identifiable Artefact。
ConstantNode	衍生自 ExpressionNode	包含常数值的具体类型 Expression Node。
	value	常数值。
OperatorNode	衍生自 ExpressionNode	引用Operator的特定类型 Expression Node。
	+operator	关联至定义了Operator Node中数 学运算符的Operator。
	+arguments	关联至Operator Node中的数学自 变量。
OperatorScheme	衍生自 <i>ItemScheme</i>	定义数学运算符的方案。
Operator	衍生自 <i>Item</i>	Operator Scheme中的数学运算 符。
	+input	关联至Operator的输入 Parameters。
	+output	关联至Operator的输出 Parameters。
Parameter		Operator的输入或输出值。

附录 A
(资料性附录)
SDMX 信息模型中关于 UML 的简要指南

A.1 范围

本附录给出UML里使用的图形符号的简短概述。本附录里使用的例子取自SDMX UML模型。

A.2 用例

为了开发数据模型，理解要求支持的函数是必要的。其在使用模型里定义。用例模型包括执行者和用例，见如下定义。

执行者的定义如下：

执行者定义了一个连贯的角色集，当系统的用户与之相互作用时可以扮演这些角色。执行者的实例可以由个体和外部系统扮演。

执行者描述成如图A.1的线条人。

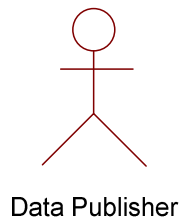


图 A.1 执行者

用例的定义如下(见图A.2, A.3)：

用例定义为一个用例实例集，其中每一个实例是一系列系统运行的动作，产生一个针对特定执行者的观测结果值。

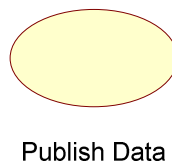


图 A.2 用例



图 A.3 执行者和用例

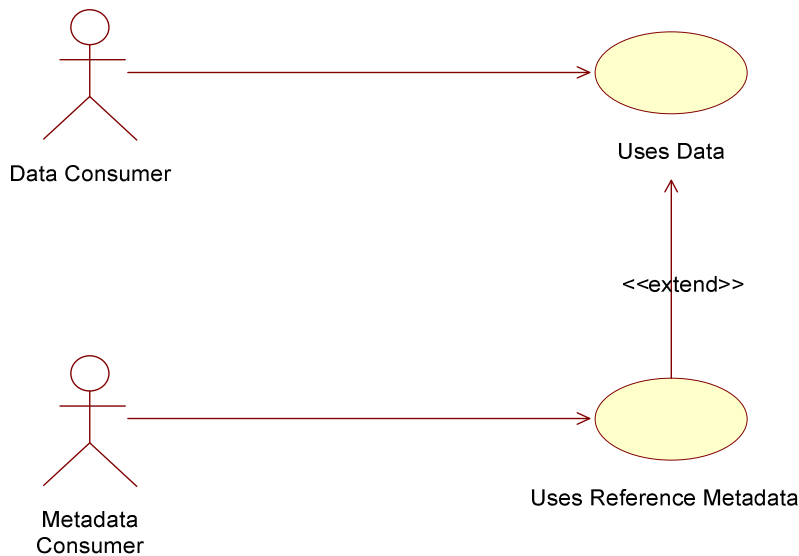


图 A. 4 扩展用例

扩展用例是一个用例可选择性地扩展一个用例，其独立于使用用例。关系图中的箭头指向它自己的扩展用例。如图A. 4，Uses Data用例被Uses MetaData用例选择性的扩展。

A. 3 类和属性

A. 3.1 概述

类是用户感兴趣的东西。在实体—关系模型(E-R模型)里，与之等价的名称是实体和属性。事实上，如果使用UML仅作为为数据构建模型的方法，则类和实体没有什么不同。

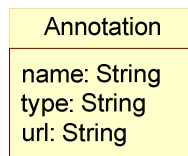


图 A. 5 类及其属性

图A. 5中，被分成三个部分的矩形代表类。上部分是类的名称，第二部分是属性，最后部分是操作。仅第一部分是必填的。类的名称是Annotation，它属于SDMX-Base包。在包里，通常把相关的类(类，用例等)分组。Annotation有3个string类型的属性—name, type, 和url。属性完整的标识符包括它的类，如名称属性是Annotation.name。

注意，依照惯例，类名称使用UpperCamelCase—单词是连起来的，且每一个单词的首字母大写。属性用lowerCamelCase—第一个单词的首字母不大写，而其他的单词首字母大写。

A. 3.2 抽象类

这里我们使用抽象类(见图A. 6)，因为它是对类进行分组的有效途径，避免了用大量关联线绘制复杂的图表，但不能预见的是：抽象类可用于其它目的(如它总是作为其子类之一使用)。本附录中图上的抽象类名称使用斜体，背景为白色。



图 A.6 抽象类和具体类

A.4 关联

A.4.1 概述

就是E-R模型里的关系。相比E-R模型，UML模型可以赋予关联更多的含义。而且，UML标记法是固定的（如，绘制关联图的方式不变）。E-R图有许多绘制技术，在E-R图里的关系有许多形式，取决于于使用的具体的E-R标记法。

A.4.2 简单关联

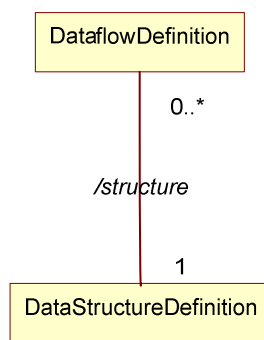


图 A.7 简单关联

这里DataflowDefinition类有一个到DataStructureDefinition类的关联。图A.7显示：DataflowDefinition的仅关联一个DataStructureDefinition（1），DataStructureDefinition可以关联多个DataflowDefinition（0..*）。有时候，可以对关联进行命名以提供更多的语义。

在UML里可能指定一系列“多样性”规则，最常见的一些是：

- 0 个或 1 个 (0..1)；
- 0 个或多个 (0..*)；
- 1 个或多个 (1..*)；
- 多个 (*)；
- 未特别指出的 (blank)。

A.4.3 集合

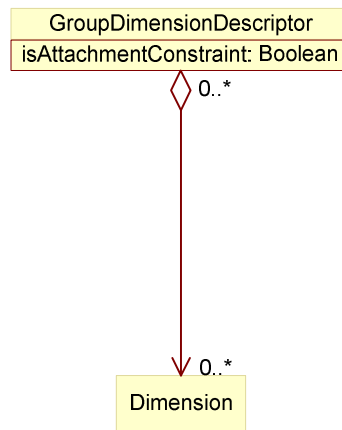


图 A.8 一个简单的集合关联

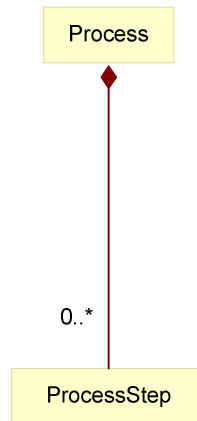


图 A.9 一个复合集合关联

有集合关系的关联表示一个类是其它类的一个从属类（或一部分）。在集合关系里，有两种集合类型。简单集合（图A.8）中，子类实例可以独立于其父类，在复合集合（图A.9）中，子类实例的生命周期依赖于其父类实例的生命周期。对于简单集合来说，这种关联在SDMX信息模型中通常也是关联类的一个参数。

A.4.4 关联名称和关联终结（角色）名称

对关联进行命名是有用的，因为它赋予模型更多的语义，即关联的目的。使用两种（或更多）关联将两个类结合起来是可能的，在这种情况下，对关联目的命名非常有用。因为赋予模型更多的语义即关联的目的。两个（或更多的）关联把两个类连在一起是可能的，在这种情况下，命名关联的目的是非常有用的。图A.10中CategoryScheme和Category（图中）间的简单集合关联称为/items（这意味着它派生自上级类之间的关联—在图中派生自ItemScheme和Item间），另一个Category之间的简单关联成为/hierarchy。

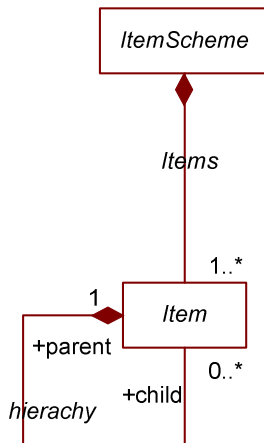


图 A. 10 关联名称和终结名称

此外,为了给出更多的语义,可以赋予“关联终结”角色名称——如在一个树状结构关联里的父和子。角色使用前面带“+”的角色名称表示(如图A.10所示,关联是指一个Item可以有0个或1个父Item以及0个或多个子Item)。

本模型中,对于具体类之间的关联优先使用角色名称,对于抽象类之间的关联优先使用关联名称。使用关联名称是因为这对显示两个子类之间的物理关联有用,这两个子类继承了其上级类之间的实际关联。在UML中,使用关联名称显示这种物理关联,而不是使用角色名称。这些内容将在后面的“派生关联”部分中讲到。

注意通常只在关联关系的一端使用角色名称。

A. 4. 5 导向性

通常关联是双向的。对于一个概念上的数据模型而言,除此之外,无需给出更多语义。

然而,UML仅允许在一个方向使用标识以表达导向性。此模型中这种导向性特征用于表示引用。换句话说,在关联中的可导向端的类引自不可导向端的类。通常,这与在XML方案中的方法一致。



图 A. 11 单向关联

图 A. 11 中,可以由 A 导向 B,但是使用这种关联时无法由 B 导向 A。

A. 4. 6 继承

有时在上级类中将通用属性和关联进行归集有用。当多个类与其他类共享同样的关联,同时共有多个属性(但不是所有的),归集是有用的。在上级类中,继承用一个三角形表示。

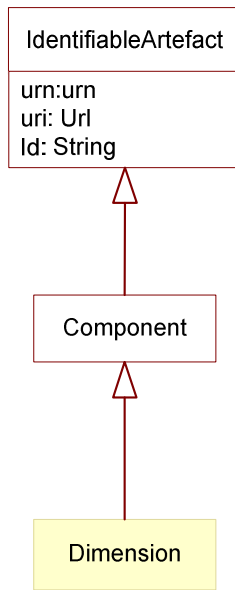


图 A. 12 继承

图A.12中，Dimension派生自Component，Component派生自IdentifiableArtefact。Component和IdentifiableArtefact均为抽象上级类。在继承树状图中，Dimension继承了所有上级类的属性和关联。注意，上级类可以是具体类(即具体类独立存在同时存在于其子类之一的环境中)，或者是抽象类

A. 4. 7 派生关联

在关系图 A. 13 中，显示子类的关联性很有用。在这里，子类继承上级类，子类之间的关联派生自上级类之间的关联。派生关联用在关联名称前加“/”表示，如/name。

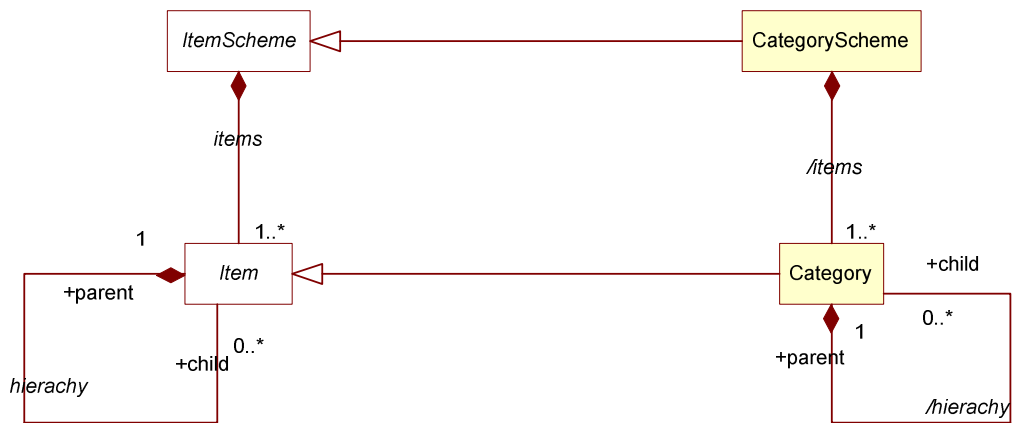


图 A. 13 派生关联

参考文献

- [1] GB/T 7408 数据元和交换格式 信息交换 日期和时间表示法
-