企业数据架构建模

背景

- 不同于书本和培训课程中所提的单例模式，真正的企业具有非常复杂的数据架构。
- 大多数数据将会存于大型遗留或包系统中，数据结构的细节对于这些系统来说可能是不可见的。
- 其他数据将存于电子表格和个人数据库（例如 Microsoft Access）中，且可能对于 IT 部门或高级业务数据管理员来说是不可见的。
- 一些关键数据可能存于由服务供应商或业务合作伙伴维系的外部系统中。

复杂数据架构

- 随着您对复杂数据架构的探究，就会逐渐接受两个现实
  - 您很少控制的了高级业务数据概念实现的方式。数据很可能是高度分散的，并且常常在质量方面缺乏足够的控制。
  - 大部分数据在大量系统中进行复制，并且在质量、格式及含义上出现重大变更。一些由企业应用程序集成（Enterprise Application Integration，EAI）技术或精心的的业务流程进行维护的副本，也许是好的（但很可能不完善）。大部分仅仅由临时的批量传输或受迫并破裂的人工流程维护的副本是很差的。组织及业务流程的冲突或简单的信任上的失败可能会阻碍常识的增长。

企业数据架构和各种组件组合

- 这些条件有几个重要的结果。例如，当计划，如客户关系管理（Customer Relationship Management，CRM）和业务智能（Business Intelligence）需要通过各种各样的来源来合并数据时，差的副本也许会使得业务或技术问题恶化。一些组织在端到端流程中利用各种遗留系统。业务或 IT 都可能会进行改变以简化业务流程，流水化数据流并减少复制。
- 尽管建模为解决这些难题带来好处，但是传统的建模方法不能解决这些难题。它们会建立要么过于详细以至于无法使用的模型，要么建立不够详细的模型，并且他们没有着重于企业数据架构和各种组件整合的难题。

UML与企业数据架构建模

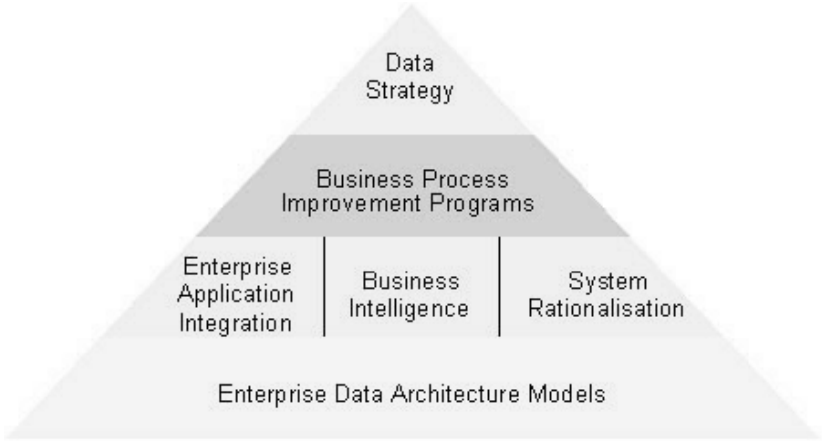
- 我们相信用企业级观点来创建强有力的、简单并有效的数据结构模型是很重要的: 一组被称为“企业数据架构”的模型。
- 使用基于统一建模语言（UML）的方法，可以满足企业数据架构建模的真正需求。

数据架构是什么?

- 企业的信息系统架构有许多相关的方面，包括应用程序、硬件、网络、业务流程、技术选择和数据。如图 1 中所示，数据架构是一组分层的模型，为战略性的计划提供坚实的基础，如：
  - *数据策略（Data Strategy）*，概括了为改进集合及数据使用的业务目标。
  - 业务流程改进。
  - 对新的变更系统的未来的决策。
  - 整合、数据存储及报告计划。

企业数据架构模型

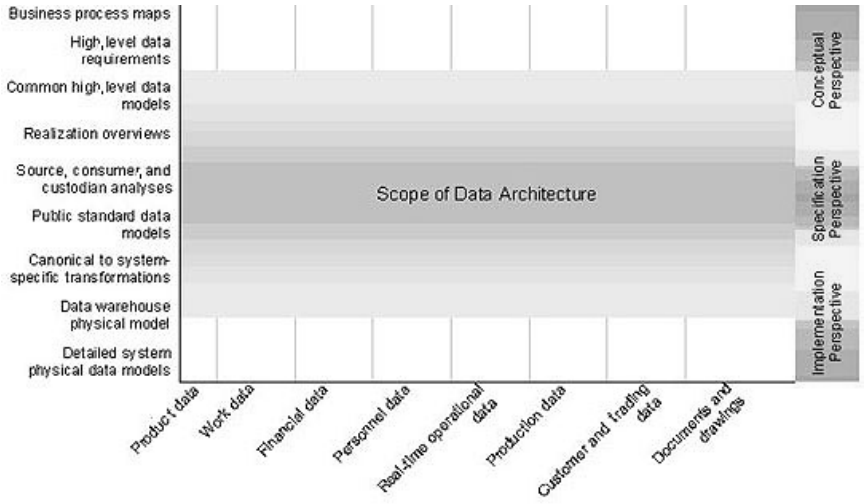
支持各种公共的 IT 和业务改进计划



数据架构不是一组单个系统的详细模型

- 数据架构不是一组单个系统的详细模型，因为它们不能传送用来满足以上需求的所需要的“大图片”信息。而且它不仅仅是业务流程和系统范围的顶级模型，因为它们没有包含足够的细节以回答实际的问题。

数据架构图



数据架构图详细解析

- 它说明了范围和数据架构环境。企业的主要数据领域映射到其中一个轴，各种类型的模型映射到另一个轴，范围从高度着重业务的模型到详细的系统架构。完整的数据架构的范围呈现为跨图表中央的带状。说明了哪个模型用于企业中哪个数据领域，完整的数据架构是横跨中央的带状。
- 水平分组是按照企业到企业区分的，上面的那些代表典型的集合。
- 在右侧边缘的带不是“图”的部分，但显示了模型如何映射到标准的基于 UML 方法（如 Rational Unified Process， 或 RUP）的三级透视图上。

- 除了利用该模型来阐述数据架构工作的范围，您还可以利用它来建立知识的当前状态和正在进行或计划着的活动的范围的映象。简单地在适当的交点绘制现有的或计划着的建模工作。您还可以通过颜色来指示模型的状态或有效性，这是很有用的。
- 数据架构图描述了“什么”组成了数据架构。支持它的数据策略和计划阐述了“为什么。”单个的模型说明数据是什么、在哪里，以及什么时候由谁如何改变的。

哪些模型构成了数据架构?

- 数据架构主要由四级模型定义的。
- 通常，只有在业务流程发生重大变更时，高级数据模型才会变更，但其他的模型将存在于各种各样的版本中，代表“目前”的结构和一个或多个“将来”的进展。

高级数据模型

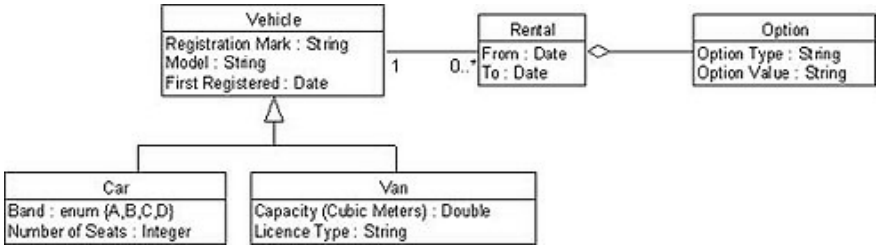
- 顶层是一组高级数据模型，用概念性观点描述业务数据，独立于任何当前实际系统的实现。每个高级数据模型（high-level data model，HLDM）包含：
  - 主要数据项（业务实体）及其关系的通用（规范的）UML 类模型。
  - 业务属性的超级，包含对这些属性含义（语义）、标准化格式（语法）和普遍制约的描述。

数据模型与实体

- 因为这些是数据模型，所以它们不会包含类方法，尽管若业务对象有责任管理其他结构的话，对这些方法进行概括是适合的。
- 模型应包含所有业务意义的属性和定义数据结构的内容（例如，控制多样性业务规则的输入）。
- 设想一个假设的汽车租赁公司。下图显示了实例 HLDM 的部分内容，说明了业务实体“vehicle”如何拥有两个变量 —— car 和 van，以及任意一个车辆（vehicle）怎样隶属于一个或多个租赁业务（rental）。

部分高级数据模型 — 用于假设的汽车租赁业务。

- 我们的实例非常地简化，但这些实例仍旧可以说明那些技术如何能够应用于带有真实世界复杂度的实例中。而且我们还在命名类及属性方面放松 UML 的约定，以使其更具可读性 —— 例如，“Registration Mark”包含一个空格。

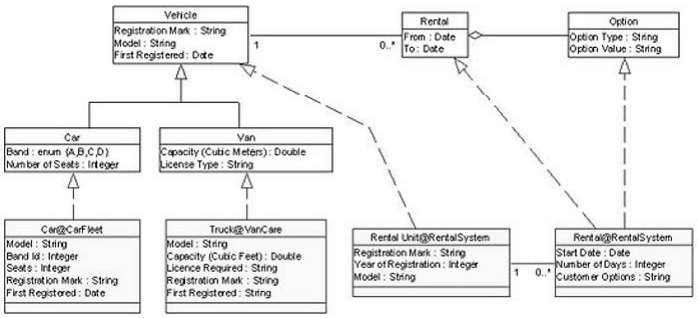


实现概述

- 下一个步骤是将 HLDM 的概念化实体和当前或计划系统的真实关键数据对象之间的关系模型化，说明了真实对象如何实现概念实体。同一数据项的不同实现之间的关系和跨多种系统扩展变更的方式将在后面进行模型化。
- 此处的关键是着重于“可见的”系统的数据结构 —— 例如，由用户接口暴露出的数据结构、报告及数据接口。这也许和物理的数据结构不同，但不重要。高度可定制的包可能内含复杂的元模型，但所关心的内容是依照业务的系统实例化。出于历史原因，旧的遗留系统也许会具有不可思议的物理结构，而且外部服务的实现细节可能会完全地隐藏在接口后面，但在这两种情况下，您的关注点将会在可视化结构上 —— *逻辑系统实体*和他们的属性。

部分实现模型

- 本图显示了简单汽车租赁 HLDM 是如何由三个系统进行实现的：CarFleet（内部队伍管理系统）、 VanCare（用于支持篷车队外包维护的外部系统）和 RentalSystem（主要的租赁控制系统）。
- 说明了三个系统中的关键数据对象如何实现来自于高级数据模型的概念化实体（呈现黄色）。



映射关系

- 对于本模型，UML 实现关系是关键。颜色和物理布局可以带来好的效果和一致的命名方案，如这个显示出的图应该能标识出逻辑系统实体及其宿主系统。
- 在概念上的实体的和真实的实体的结构或含义有所不同的地方，当可以直接将关系进行映射（如上图所示）之前，一般化和聚集关系是用来分解类结构的。甚至当 HLDM 作为元模型并且实现模型是具体的时候该方法也可以使用，反之亦然。

来源及使用者模型

- 模型的下一层显示了同一数据项的不同实现之间的关系、如何在不同系统上扩展变更，以及不同数据元素的组织的管理人。

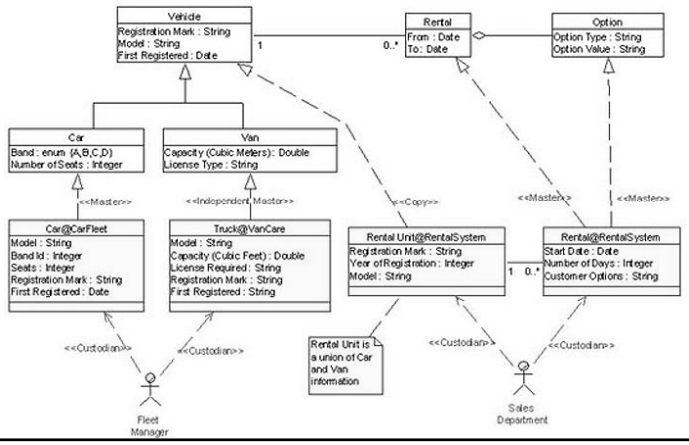
使用构造型

- 除了关注点在识别角色、起源及每个数据项的发展上意外，模型类似于实现概述，利用下面的构造型：
  - <<Master>> 标识已经过协议的原版数据源。
  - <<Use in place>> 和 <<Update in place>> 标识一个能够直接通过现有接口使用另一系统数据的位置。注释将解释其如何工作的。
  - <<Copy>> and <<Updates Copy>> 标识出一个系统得到了另一系统的规则或不规则的数据（或者更新列表）的副本，及该副本是否未修改或被接收系统修改了。注释中将描述计时及相似的问题。
  - <<Independent master>> 标识出在哪里系统不是原版的，并且在理论上应具有原版数据的副本，但由于流程不是足够地确定，所以第二数据集出现分歧。
  - <<Custodian>> 标识出数据项和组织或角色（显示为业务角色，和适当的数据类有依赖关系）的保管关系。
  - <<Uses>> 标识出重要的跨组织的数据的使用。

- 在需要对不同属性用不同方式进行处理的地方（例如，一个实现对类的一些属性是原版的，另一个类对其他属性是原版的），高级数据模型应通过两个或多个分离的类对那些属性进行建模。来源及使用者模型（下图）能够清楚地说明不同的责任及他们的来源。

来源及使用者模型

- 添加描述不同实现如何相关及它们如何与不同组织角色相关的信息（绿色）。



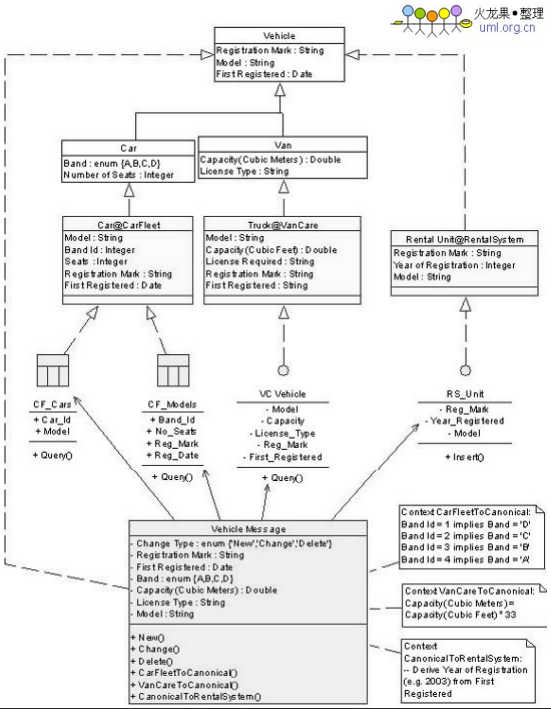
迁移及转换模型

- 模型的最后一层描述了执行系统的数据如何在系统间移动时进行转换。它们表现在：
  - 系统*接口*的物理类及属性结构（等同于数据库结构，在其中直接数据访问是最好或唯一的选择）。该模型还将显示带有接口机制（如 EAI 集线器或干线）的 HLDM 的实现。
  - 不同物理数据结构之间的实现关系。
  - 属性级的转换规则，用目标约束语言（Object Constraint language，OCL）编制而成。
  - 接口驱动器、约束和计时规则，通过交互或时序图进行建模。

- 扩充我们的汽车租赁实例，假设我们想通过 EAI 来保持 RentalSystem 中列出的 Hire Unit 保持最新，提取、合并，并转换两个源列表。下图中的“将来”的模型描述了物理接口和转换规则，包括 EAI 消息中数据的正则结构。
- CarFleet 有一个由两个主要表格组成的基于数据的接口，Vancare 有一个程序化的接口（例如，对象模型或 Web 服务），Rental System 也一样，包含一个接收更新的 Insert() 方法。

转换模型

- 添加说明数据在系统间移动时如何转换的细节。

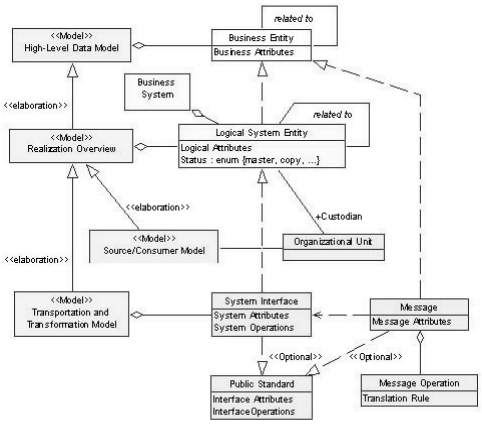


公共标准

- 公共的或行业的标准有两个任务：
  - 它们为 HLDM 或在 EAI 干线和外部接口内的 HLDM 的实现形成基础。
  - 它们会确定外部接口或一些物理系统的数据结构，因此描绘出在接口中进行转换的物理数据结构。

元模型

- 图中的元模型展示了设计中各种模型及其组件之间如何相关



使用并开发数据架构

- 数据架构有很多用途。它帮助您处理数据如同它真的在由业务使用，并且如果您想要开发并实现支持数据策略的治理，它是其中关键的部件。它还应该用于指导跨系统的开发，例如企业应用程序整合（EAI）、公共报告和数据存储计划。

- 虽然我们对数据架构的阐述是按照“自顶向下”的，但是数据架构通常按照“由中间开始”进行开发，从特定系统接口的数据需求和合理化实行开始，并不按照严格的自顶向下的流程和信息需求分析。这种方式允许数据架构开发以满足不带有难于管理的依赖性的具体的战术战略需求，并向基于分离的自顶向下和自底向上建模运行的数据分析提供交叉校验。

- 数据架构业务对整个企业从来不是“完整的”。虽然如此，它提供了一种一致的方法和用于建模活动的环境。然而，随着数据架构的成熟化，采取一些工作来“填补空白”是适合的。
- 模型，尤其是来源及使用者模型，将通过确定目标数据是否包含于单个系统，由良好定义的接口和流程进行维护，或在一些（潜在地不一致的）源中间进行扩展，来支持目标业务流程的验证

改进数据架构：数据策略

- 将“目前”的数据架构进行建模非常有用，它能够很确定地显示出哪里是次佳的。然而，如果您想要进行提高，就需要有比好的模型多得多的东西。围绕改进数据集合、使用和治理的大部分问题是非技术的。

IT 部门，及业务经理，需要开发若干内容

- 设定企业如何收集、管理并使用数据的原则。
- 包含“目前”的和“将来”的模型的数据架构。
- 数据架构的治理规则和变更控制流程，由 IT 和适当的业务代表共同管理。
- 在每个业务领域内的数据管理规则：
  - 存储什么数据。
  - 谁负责数据的收集和质量。
  - 谁控制，谁管理
  - 存储多久，将来如何安排或归档。
  - 谁可以使用，及如何向常规用户组之外的用户公开。
- 关于信息和相关风险分类的方案，以确保定义恰当的安全方法。

数据策略需要建立在清晰的意见一致的原则之上

- 不论在哪里，数据的输入必须简单且数据能准确地反应情况，还要以一种对输入输出有效的且可用的格式。
- 如果数据具有已知且编制了的用途和值，就应收集。
- 对于那些对数据有合法业务需要的数据应是易用的。
- 数据获取、验证和处理的流程不论在哪都应是自动的。数据只应输入一次。
- 在整个企业范围内，更新所给数据项的流程应是标准的。
- 应尽可能准确完整地记录数据，利用最广博的来源，使其与原始内容尽可能接近，在最初的时候将其变成电子格式，并采取可检查可跟踪的方式。
- 数据收集和共享的费用应最小。
- 企业，而不是任何个人或业务单元，拥有所有数据。
- 每个数据源必须有确定的管理人（业务角色）负责数据的精确性、完全性和安全性。
- 防止对数据进行未授权的访问和更改。
- 除非有实践的原因需要进行复制，否则不能够对数据进行复制。在此情况下，一个源必须明确地作为原版，要有健壮的流程确保每一步的副本，并且不能修改副本。
- 数据结构必须在严格的变更控制下，以便于可以适当地管理各种业务和系统牵连的变更。
- 无论什么时候，对公共数据模型采用国际、国家或行业标准。在不可能采用时，开发组织的标准来替代。

结束语

- 对企业数据架构的文档化的理解是许多公共 IS 和业务改进计划的必要先决条件。适当的模型与详细的系统模型和高级业务模型截然不同。本文概述了一组有助于满足这些需求的 UML 模型和技术。