

基于本体的元数据应用

Ontology-based Metadata Application for Digital Libraries

刘炜

上海图书馆 200031

李大玲

南开大学国际商学院图书馆学系 300071

夏翠娟

华东师范大学信息学系 200062

摘要: 本文简要分析了知识本体对于数字图书馆的作用、功能及其实现方式, 论述了知识本体概念的来源、含义和目前的研究与应用进展, 以及知识本体与传统的图书分类法和叙词表的关系; 着重阐述了知识本体对于元数据方案所起到的补充和高层互操作的作用, 以及建立知识本体的一般流程和方法; 介绍了各种知识本体语言和工具起源、特性和发展情况, 并对本体工具的性能和特性做了比较分析。

Abstract: An ontology is a formal specification of a conceptualization, usually related to a specific domain of knowledge like library and information science. The metadata application profile along with the documentation of its abstract model can be thought of a primitive ontology of a specific implementation. Classification themes and thesaurus, which have been used for a long time in the library and information arena, are also a source of formal ontologies. After formalization processing and encoded with standard ontology languages, these kinds of concept systems can be very useful to establish a metadata service based on ontology services and fulfill the high level interoperability of digital libraries. And many languages and tools used to establish formal Ontology are introduced and comparatively analyzed in this paper.

作者简介:

刘炜, 上海图书馆数字图书馆研究所所长, 研究员, 从事数字图书馆、元数据、知识本体、异构信息系统的互操作研究等。著有《数字图书馆引论》一书, 参与编写《DC 元数据》。

Email: wliu@libnet.sh.cn

李大玲, 南开大学国际商学院图书馆学系在读博士

夏翠娟, 华东师范大学信息学系 2002 级硕士

知识本体 (ontology) 本来是哲学中的一个概念。近年来, 知识本体在人工智能领域引起引起研究人员的兴趣, 并越来越多的应用在万维网信息的表示、组织与管理上, 一个重要原因就是当前对基于网络的知识共享和知识交换存在巨大期望和需求。随着各种已有的和将要开发的 Web 应用的不断增多, 人们希望不同的系统能够共享某个共同公认的知识库或者词汇 (术语) 集合, 提供统一的领域模型, 这样就极大地促进和实现了不同系统间的数据或知识共享、交换和重用。

数字图书馆作为一个分布异构环境中的知识体系，为知识的语义理解、计算机理解、计算机和人进行交互提供了应用框架和实现途径。虽然知识本体在人工智能、知识表示中已经被广泛的讨论和实践，但在我国图书馆情报界及数字图书馆研究领域中的研究仅仅局限在概念的界定和初步的理论研究层面上，对于知识本体的方法论、编码语言、构建工具、具体构建缺乏实践。本文在讨论知识本体在数字图书馆中的具体作用的基础上，对知识本体的语言、方法论、构建工具进行介绍和分析，希望能够为数字图书馆中知识本体的建设提供一个初步的理论和实践的基础。

数字图书馆功能需求

解决分布式网络环境下系统或资源间的互操作问题是数字图书馆技术的核心内容。“异构”是指系统或资源在结构上的不同，互操作是指系统或资源之间的兼容性或关联关系。万维网是目前最大的开放的分布式网络，可以看成由无数三层结构应用¹组成的大型资源库群（repositories）。这些资源库群是彻底异构的，从数据结构、操作系统，到数据库系统，到应用系统；从命名方式，到数据格式，到结构模型，到用户界面，都有可能完全不同，目前还没有多少标准规范能够对这个各个层次的异构进行适当的约束，数字图书馆在这个方面尚缺乏完整解决方案，而且解决方案也不是唯一的。从体系结构上来看，“语义万维网”和“Web服务”技术正在形成一套异构系统互操作问题完整的解决方案。

元数据提供了数字图书馆的语义基础，使资源有了基本的微观结构，但是元数据并不能完全解决信息系统的语义异构问题，包括资源采用不同元数据方案所造成的微观结构的异构问题以及资源对象之间存在的复杂的关联关系，知识本体在某种程度上可以看成是“元”元数据，信息系统中不同实体对象可能采用不同的元数据方案，不同的实体对象之间的关联关系非常复杂，知识本体能够对这些情况进行很好的描述，从而为信息的组织、管理以及检索、查询提供模型和方法。

异构是普遍存在的，元数据对于资源描述的特殊性和一般性的矛盾与生俱来，是其本身无法克服的。或许随着标准化的进程，DC元数据等少数元数据格式将占据主导地位，然而永远不可能统一到仅有少数几种格式。许多专业或专门领域仍然会有大量的元数据方案，这些元数据方案可能局限于一个狭小的领域，其本身就是一种领域本体，但是只有专业的元数据对于专业的应用才是最合适的，与学科外其他领域的互操作性考虑是次要因素。在网络环境下要联接这些“信息孤岛”，必须有某种程度的互操作解决方案，而且最好是标准的解决方案，这就需要在元数据之上再建立某些机制，来灵活地实现信息系统之间的互操作。知识本体的本质就是领域知识的共享和重用，标准化和形式化的领域本体能够为信息系统之间的高层互操作提供很好的工具。

什么是知识本体

知识本体（Ontology）被赋予了太多的含义，从抽象的哲学思辨，到实用的计算机推理。牛津英语辞典里对“Ontology”的解释是“对于存在的研究或科学（the science or study of being）”，人工智能领域经常引用 Gruber 在 1993 年的定义“概念体系的规范”（specification

¹ 三层结构应用指由“浏览器-Web 服务器-数据库服务器”组成的应用体系，俗称动态网页技术，相对于将所有发布内容均以 html 文件形式存放于 Web 服务器的 B/S 两层结构而言。

of conceptualization)², 1998年 Studer 等人在这个定义的基础上对于本体的特点给出了一个较为明确的解释:“知识本体是对概念体系的明确的、形式化、可共享的规范说明”(见参考文献)。直观地,我们可以把知识本体看成是“领域知识规范的抽象和描述,表达、共享、重用知识的方法。”

知识本体作为领域概念及概念之间关系的规范化描述,这种描述是规范的、明确的、形式化的,共享的。“明确”意味着所采用概念的类型和它们应用的约束实行明确的定义。“形式化”指知识本体是计算机可读的(即能被计算机处理);“共享”反映知识本体应捕捉该领域中一致公认的知识,反映的是相关领域中公认的概念集,即知识本体针对的是团体而非个体的共识。知识本体的目标是捕获相关领域的知识,提供对该领域知识的共同理解,确定该领域内共同认可的词汇,并从不同层次的形式化模式上给出这些词汇和词汇间相互关系的明确定义。

如果把每一个知识领域抽象成一套概念体系,再具体化为一个词表来表示,包括每一个词的明确定义、词与词之间的关系(例如“用”“代”“属”“分”“参”关系等)以及该领域的一些公理性知识的陈述(例如“所有的期刊论文都是出版物”)等,并且能够在这个知识领域的专家之间达成某种共识,即能够共享这套词表,所有这些就构成了该知识领域的一个“知识本体”。最后,为了便于计算机理解 and 处理,需要用一定的编码语言(例如 RDF/OWL)明确表达上述体系(词表、词表关系、关系约束、公理、推理规则等)。在这个意义上,知识本体已经成为一种提取、理解和处理领域知识的工具,可以被应用于任何具体的学科和专业领域。实际上图书馆领域很早就在进行类似的工作了,主题词表、分面分类的思想即是初始的萌芽,今天能够通过严格的形式化之后借助计算机的强大处理能力,可以说对网络知识的“整序”已经呈现出令人激动的曙光。

对某个知识领域每个人的认识从内容到形式都可能是不一样的,通用的高层知识本体(Common Ontologies)常常从哲学的认识论出发,其概念的根节点往往是时间、空间、事件、状态、对象等抽象术语,而且不一定需要形式化;领域本体(Domain Ontologies)专注于解决领域知识的抽象,较为具体,容易进行形式化和共享;术语本体(Terminology Ontologies)常常表现为一个词表,概念关系的抽取较为随意和简单,不严格要求,甚至可以没有概念定义,例如著名的 WordNet 本体;形式本体(Formal Ontology)对于概念术语的分类组织要求较为严格,需要按照一定的分析原则和标准,明确定义概念间的显性、隐性关系,并明确各种约束、逻辑联系等,这类本体常常由术语本体发展而来,但却与术语本体没有截然的界限;另外还有表现本体、任务本体、方法本体、混合型本体等等。

需要注意的是在图书馆界知识本体并不能简单地等同于传统图书情报领域的叙词表,它也不是一个孤立的技术,从宏观上讲与元数据一样是数字图书馆的一个重要组成部分和重要技术和工具,从微观上讲可以解决在一个系统中的它可以理解、表达为一组概念(如实体、属性、过程)及其定义和相互关系。知识本体一般包含某一领域的概念网络并通过一种“属性-值”机制来描述每个概念的关键特性。该网络可以是有向的也可以是非有向的。它也可能是某一特定类型的网络,即一种概念层次(树)。概念之间的关系可通过附加逻辑语句加以描述。目前研究人员已经定义了不同领域的知识本体模型,并且有些知识本体模型已经投入了实际应用,这些知识本体模型的描述方式、形式化程度、建模目标等都各不相同。

如上所述,某个具体领域的知识本体不可能是唯一的,形式化方式手段也可以不同,但是不同的知识本体必须通过某种机制保证交换和映射的顺利进行,形式化的方式也需要标准化,这就是知识本体语言的作用。

² 见: <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html> (2004/4/24)

知识本体的作用

知识本体的作用可以从两个角度来理解：应用于所有领域知识规范化的一般作用和应用用于数字图书馆建立语义模型的特殊应用。从一般领域应用的角度来理解，知识本体主要有以下作用：

在人、机器（表现为软件代理）以及人与机器之间共享对于信息及结构的共同理解

这是知识本体开发最基本的一个目标。采用术语和关系来编码领域假设。举例说明，假设几个不同的 web 站点包含医药信息或者提供医药电子商务服务。如果这些站点之间共享和发布他们共同使用的术语的知识本体，那么计算机代理就可以从这些不同的站点中抽取并集合信息，用这些集合的信息来回答用户的查询请求或作为其它应用的输入数据。

实现一定程度的领域知识的重用

促进领域知识的重用推动了知识本体的研究。下面给出一个知识本体在领域知识重用方面的例子，比如有多个不同领域组织的模型均需要表示时间这一概念，时间的表示包括时间间隔的概念和定义、时间指针（points）、相关的时间测量方法等。如果这些领域和组织中有一个组织详细开发了满足要求的知识本体，其他领域和组织就可以很方面简单的把这个知识本体应用到自己的领域中来。此外，如果需要开发一个大型的知识本体，可以通过集成描述大型知识本体某些部分的多个现成的知识本体来实现。也可以通过重用诸如 UNSPSC 的通用知识本体，并对这类知识本体进行扩展来满足我们对感兴趣领域的描述要求。

知识本体可以明确领域假设，使领域公理得到明确描述从而达成共知

通过知识本体可以明确领域假设，这些领域知识的明确说明对于要了解该领域的新用户了解该领域中的术语非常有用。如果关于领域的知识发生变化，通过知识本体可以非常容易的改变关于该领域的假设。如果关于领域的假设被隐藏到了程序语言代码中，则这些假设非常难以发现和理解，更难修改，特别是对那些不懂程序的人而言更是如此。

对于领域知识进行分析、明确，并使其形式化

一旦明确说明了一个领域中的各种术语，就可对领域知识进行分析。当要重用现有知识本体和扩展现有知识本体时，对术语的形式化的分析就体现出它的重要价值。通常而言，一个领域中的知识本体的目的不局限于构建它时的目的，而是为了领域知识的重用。问题解决方法、独立于领域的应用和软件 agents 把知识本体和知识本体生成的知识库作为数据来使用，在 web services 中作为单个 service 来使用。

一个应用了知识本体和元数据的数字图书馆系统，资源的组织在微观层面都是依据各种规范的元数据方案，资源之间的宏观联系依据知识本体所形式化的联系模型，知识本体使各类元数据方案联系成一个立体的知识网络，并能使资源按照知识网络中的不同属性，或同一种属性的不同编码体系，呈现出规范有序的知识地图，供浏览或检索。并且还可以通过开放某些标准的元数据接口，或提供某些可供各式转换的映射表，向某些登记系统（例如 UDDI）进行注册，从而达到更大范围的互操作。在数字图书馆中知识本体的主要作用有如下一些：

提供与描述型元数据有关语义描述的知识地图

元数据是关于数据的数据，虽然在不同的领域中定义不尽相同，但是其基本的含义却是对资源对象固有属性的描述，无论对元数据怎样分类，总体来说，可以认为元数据都是描述性的。

元数据提供了数字图书馆的语义基础，使资源有了基本的微观结构，但是元数据并不能

完全解决信息系统的语义异构问题,包括资源采用不同元数据方案所造成的微观结构的异构问题以及资源对象之间存在的复杂的关联关系,知识本体在某种程度上可以看成是“元”元数据,信息系统中不同实体对象可能采用不同的元数据方案,不同的实体对象之间的关联关系非常复杂,知识本体能够对这些情况进行很好的描述,从而为信息的组织、管理以及检索、查询提供模型和方法。

异构是普遍存在的,元数据对于资源描述的特殊性和一般性的矛盾与生俱来,是其本身无法克服的。或许随着标准化的进程,DC元数据等少数元数据格式将占据主导地位,然而永远不可能统一到仅有少数几种格式。许多专业或专门领域仍然会有大量的元数据方案,这些元数据方案可能局限于一个狭小的领域,其本身就是一种领域知识本体,但是只有专业的元数据对于专业的应用才是最合适的,与学科外其他领域的互操作性考虑是次要因素。在网络环境下要联接这些“信息孤岛”,必须有某种程度的互操作解决方案,而且最好是标准的解决方案,这就需要在元数据之上再建立某些机制,来灵活地实现信息系统之间的互操作。知识本体的本质就是领域知识的共享和重用,标准化和形式化的领域知识本体能够为信息系统之间的高层互操作提供很好的工具。从而提供与描述型元数据有关语义描述的知识地图。

提供资源库领域知识的规范描述

当前不同的机构和部门根据自己的需要建设了形式多样,内容各异的资源库,这些资源库包括各种教育资源库、法律资源库、地方志资源库、传媒资源库、专业技术资源库等等,其中教育资源库涵盖了从小学、中学到大学、从个人教案到国家级的资源库,种类繁多。如果能够充分利用这些资源库将会促进各行各业的知识共享和迅速发展,降低成本,避免重复建设。但是当前实际情况却是这些资源库一般没有完整的结构、存在重复建设、数量庞杂、形式不规范等问题。

如果我们把知识本体引用到资源库的建设中,通过知识本体对资源库的领域知识进行识别和规范描述,达成领域内关于知识和概念及概念关系之间的共识。这样引入了知识本体点各种资源库就能够实现真正的重用和共享,能够解决资源库建设中存在的问题。

提供元数据映射方案,集成到数字图书馆体系中的元数据服务中,成为协议的一部分

在数字图书馆中,存在着各种各样的信息系统,要实现这些异构、分布的系统之间的互操作是一个难题,解决异构系统之间的互操作的一种解决方案是采用元数据。但是不同的系统中存在着不同的元数据方案,这些方案采用的标准不同,相同的术语可能存在不同的语义,不同的术语的语义又可能相关。元数据知识提供的只是这些异构分布系统互操作的一个基础,所以需要在元数据之上采用一种新的技术和方法来实现这些元数据之间的互操作,才能够实现这些系统之间的互操作问题。这种技术和方法采用高层互操作协议。这种高层互操作协议包括元数据交换协议和相关知识本体协议。元数据交换协议能够实现元数据之间的映射,但对于元数据及信息系统之间的互操作,还要在元数据交换协议基础上采用知识本体,实现这些元数据方案之间的语义映射、不同的元素之间的关系定义及规则约束,从而真正实现这些异构分布系统之间的语义互操作。

提供智能代理与信息环境之间基于语义的理解机制

智能代理指的是一种计算机技术,这种技术模仿人的行为执行一定的任务,而且在执行这个任务的时候不需要或很少需要人的干预与指导。

智能代理的主要功能有,管理个性化的信息代理库,主要可以管理用户个人资料及其个人目录下的信息库;信息自动通知,当信息用户指定了特定的信息需求之后,智能代理能够自动探测到信息的变化和更新,进而将其下载到数据存储地存放起来,同时智能代理能将该信息自动地提示给用户;浏览导航,信息用户如果愿意在网上去冲浪,智能代理能分析到该用户所感兴趣页面所属领域,并能向该信息用户建议与该领域更密切的页

面或链接；智能搜索，信息用户在网上搜索信息时，往往为搜索到的信息太少或可用度差。而智能搜索，能够根据信息用户的特定需求，进行信息过滤为用户提供更精确的搜索信息；生成动态个性化页面，智能代理能依据信息是所存放的信息动态地生成网络页面，给信息用户提供一个适宜的而友好的浏览界面。此外，智能代理还具有监督代理，协调与解决冲突等功能。

信息环境，指的是一个社会中由个人或群体接触到的信息及其传播活动构成的环境。

从上面对智能代理和信息环境的含义中，我们可以看出智能代理必须与信息环境之间实现语义理解，而不是仅仅通过关键词的匹配，才能够实现它的功能。这种语义理解实际上包括，计算机与人之间和计算机与计算机之间的相互语义理解。而知识本体在 URI、XML、RDF、RDFS 和元数据的基础上，提供领域知识的概念体系的确定，为智能代理和信息环境之间提供基于语义的理解机制。

跨平台、跨系统之间的通信中介

正如本文概念界定中指出，数字图书馆是提供异构信息系统的互操作的一种环境，在这个环境中，分布异构的不同信息系统之间可以实现互操作。而实际情况是，由于这些信息系统是彻底异构的，从数据结构、操作系统，到数据库系统，到应用系统；从命名方式，到数据格式，到结构模型，到用户界面，都有可能完全不同，目前没有多少标准规范能够对这个各个层次的异构进行适当的约束，数字图书馆在这个方面尚缺乏完整地解决方案，而且解决方案也不是唯一的。从体系结构上来看，“语义万维网”和“Web 服务”技术正在形成一套异构系统互操作问题完整的解决方案。知识本体作为语义万维网中的重要技术和工具，可以通过为不同的领域构建领域知识本体（domain ontologies），然后再在这些领域知识本体之间建设上层知识本体（upper-ontology），结合其他技术来实现这些系统的互操作，实现这些系统跨平台和系统的信息系统之间的通信。

分布环境下查询请求的语义理解

现阶段查询请求主要是通过对查询语句进行解析，解析成一个个的单词然后进行关键词的匹配，把匹配的结果按照一定的算法的进行过滤和排序提供给用户。一般没有对其语义进行解析。有的系统是通过抓取 web 页面 head 区的元数据来提供一定的相关理解，或通过内容敏感链接来查找相关的信息。对于查询请求的语义理解特别是基于自然语言的理解，和查询请求在分布环境下的分发依旧是一个难点。假设一个领域中都建设了相应的知识本体，那么基于自然语言的查询请求就可以翻译成某一个领域知识本体中公认的概念组成的查询请求，再通过知识本体的影射和互操作实现对不同领域中相关主题和不同语言的相关主题的检索。这样就实现了对查询请求的语义理解。

比如要查询“熟悉 XML 的专家”，如果按照关键词检索的方式，检索的命中结果中需要包含“熟悉”、“xml”、“专家”。如果某个专家出版了一本关于 XSLT 的书，在该专家的著录中没有包含 xml，那么这个专家就被排除在检索结果之外了。采用知识本体则能够提高命中率。看似简单的检索式涉及复杂的逻辑概念、语义和语法关系，如果一本书的主题是关于 XSLT 的，那么这本书的作者就符合检索表达式。

数字图书馆中的数据挖掘

数据挖掘是一个人机交互、不断重复的过程，专家的领域知识或背景知识的应用对挖掘过程具有补充和促进作用，经常用作引导发现过程以避免无意义的结果。另外，一般数据挖掘方法仅仅在数据库内容上产生规则，规则难以理解，领域知识或背景知识的应用可产生易理解的规则。知识本体是知识表示的一种形式，它能将领域知识表示成挖掘算法能够理解的形式。

知识本体已经被应用到数据挖掘中，这些于知识本体的数据挖掘主要有基于知识本体的

多媒体数据挖掘、基于知识本体的 web 数据挖掘、基于知识本体的数据挖掘智能助手、网格计算中数据挖掘知识本体等方面。

在数据挖掘中我们可以采用基于知识本体的智能发现助手（IDA）来协助数据挖掘。通过知识本体来形式化处理前要用到各种预处理的概念和处理、形式化演绎的各种算法、形式化处理后的一些转化和模型，从而可以根据不同的用户提供的数据挖掘的各种参数，满足用户的数据挖掘需求，有效改善数据挖掘的效果。也可以通过用知识本体来形式化明确说明多媒体和 web 数据，来把知识本体应用到数据挖掘中去。

针对上一节提到的元数据标准规范的问题，知识本体正好从某种程度上弥补了元数据的不足：

- 元数据方案不具有普遍适用性。无法克服特殊性与一般性的矛盾，而形式化的知识本体可以提供一种在元数据方案之间自动映射的机制，通过语义 Web 服务的体系架构进行实现；
- 元数据应用难以实现元数据方案本身的进化，而知识本体可以提供信息系统的其它视图，只需要通过自动或半自动的手段应用新的元数据方案；
- 元数据方案自身难以对不同知识体系、不同“粒度”的资源进行描述，而知识本体正是起到这个作用，从而实现异构资源和系统之间的语义联系；
- 单纯的元数据方案对于数字资源的整个生命周期的描述非常困难，而采用以诸如 FRBR 模型为基础的知识本体，这个问题便迎刃而解，不同生命周期的知识产权属性也非常易于描述；

除此之外，知识本体同时也在一定程度上解决了诸如灵活性和可扩展性问题，以及在资源集合层面的整合的难题。

知识本体如何实现功能

知识本体作为数字图书馆语义模型的形式化，主要功能体现在信息资源的组织和信息检索查询两个方面，如图 1 所示：

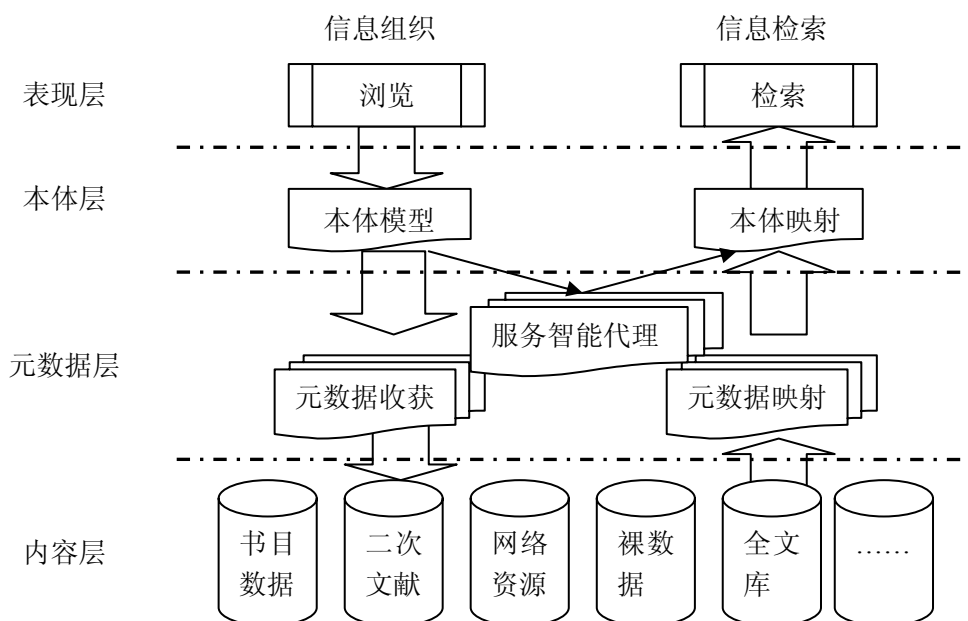


图 1：基于本体的信息系统模型

数字图书馆的资源不论是虚拟的还是实在的，不论涉及单个还是多个信息系统，其涉及的实体类型往往不可能是单一的，这些类型之间也往往具有复杂的关系，因此很难用一套平面的元数据方案进行数据组织。例如傅雷翻译罗曼·罗兰的作品《约翰·克利斯朵夫》，曾经有译林出版社 2002 年、哈尔滨出版社 2000 年、中国友谊出版公司 2000 年、河南人民出版社 1998 年、安徽文艺出版社 1990 年……平明出版社 1952 年等数十家出版社几十个版本，并且还有翻译手稿、有声读物、衍生电影、戏剧剧本和声像资料、英文原版著作等相关资料，以及傅雷和罗曼罗兰生平资料等等，这些信息不论是否存在于分布的信息库中，都应该通过一定的方法进行有效的映射和描述，但显然现有的平面的元数据方法是无法实现的，但是利用本体模型（例如用 ABC 本体模型，见图 2 所示³）却能清晰准确地揭示这些资源对象各类属性及相互关系，这种描述方式对音像出版物等多媒体资源所涉及的复杂责任关系和版权关系特别有帮助。知识本体模型原本就是对领域知识的归纳和形式化，目的就在于共享和重用，因此特别适合作为信息模型对知识系统进行描述、表达和呈现。

如果我们把图书分类法看成一种基本的简单的知识本体，一个书目数据库就可以按照分类法的层次结构组织成一个庞大的树，每一片叶子就是一本书。这样可以形成一个简单的、一维的知识导航地图。当我们同时采用分类主题词表或其它分面分类方法对资源的内容从不同的“本体”角度进行揭示，整个资源库（数字图书馆）就有了多维的导航机制。更进一步，通过不同知识本体的映射可以动态建立从一个信息库到另一个信息库的语义连接，这种连接并非预先设立的，而是“后组”的。并且如果有本体注册服务中间件或代理进行自动的翻译、映射服务，就能从很大程度上解决知识的跨库提取、动态浏览展示以及异构系统的动态勾连等问题，实现数字图书馆彻底解决异构信息检索的目标也就为期不远了。

³ 本例的资源对象关系分析把傅雷的译作当作对原作的再创作，作为具有翻译关系的“作品”（WK1）。

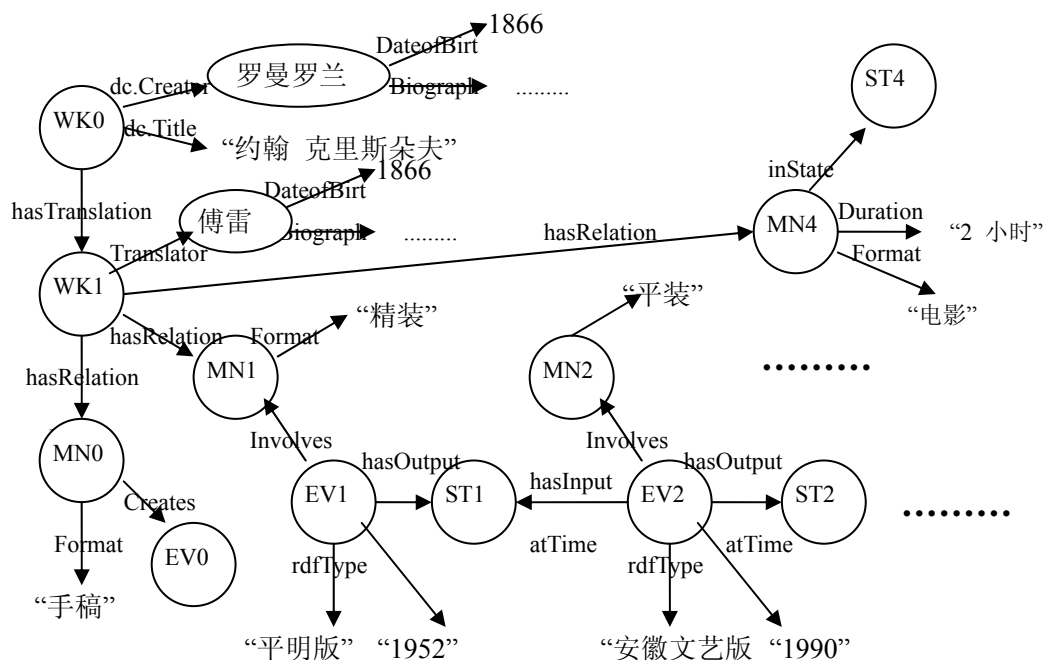


图 2: ABC 本体模型描述傅雷翻译作品《约翰 克里斯朵夫》

对于查询请求来说，知识本体的应用能够实现许多以前无法实现的查询请求，例如基于多种关联关系的查询：“何时何地何人做了什么”。并且基于查询处理中介或代理的帮助，查询提问式可以智能地处理成复合不同资源集合的规范词或者表达式形式，自动分发到不同的资源站点进行查询，同时还可以对返回结果进行基于本体的排序处理，将最终结果返回给用户。

本体描述语言

语义 web 语言源于历史上开发的多种基于 web 标准的语义描述语言，其中不少就是以描述和构建知识本体为目的而开发的。

SHOE (Simple HTML Ontology Extensions)

SHOE 是一种基于 HTML 的知识表示语言，由美国马里兰大学 (University of Maryland) 并行理解系统组 (Parallel Understanding Systems Group) 于 1996 年开发。SHOE 对 HTML 进行扩展，使其能够用 HTML 格式对知识进行表示。

HTML 并不是一种计算机能够“理解”的语言，它的功能只限于将数据表示出来供人阅读。HTML 的“知识”一些是通过自然语言，另一些是通过陈列图表等来表示，这都是人能理解的方式，计算机很难理解这些知识。

SHOE 试图提供一种对信息进行标注的方法来表示知识。其提供一套必要的标签 (tag) 将专用的语义数据加到 Web 资源当中，从而对知识进行表示。这些标签分为两类：一类是为构建各种知识本体来使用的，另一类是用来标注 web 文件。对于第一类，SHOE 的知识本体是规则的集合，其用来规定 SHOE 文档可产生什么类型的声明以及这些声明是什么意思。对于第二类，标注 web 文档的标签用

来描述一个或多个知识本体，声明数据实体，并且再 SHOE 的知识本体预先描述的规则产生关于则会些实体的声明。

SHOE 允许表示概念、概念分类、n 元关系、常量以及推理规则，其推理引擎可以通过这些推理出新知识。

“Knowledge Annotator” 工具可用于在 HTML 中嵌入知识本体信息。

XOL (XML-based Ontology-exchange Language)

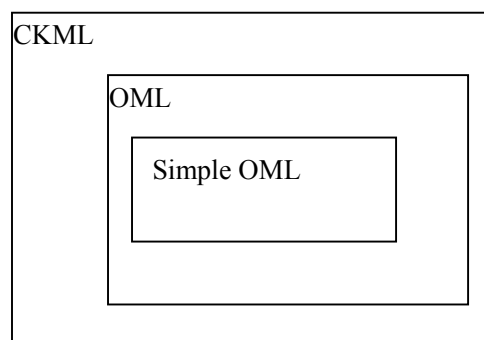
XOL (XML-based Ontology-exchange Language) 是 SRI 人工智能中心于 1999 年开发的一个 XML 化的知识本体交换语言，其从 OKBC (OKBC-Lite) 协议继承了一个小的源语子集。

XOL 起源于 BioOntology Core Group 发起的生物信息学 (bioinformatics) 知识本体交换语言的研究，该研究需要一种能表示面向对象的语义又基于 XML 语法的语言，XOL 被设计来满足该要求。后来 XOL 成为一种通用的语言，可被用于任何领域知识本体的描述和交换，因此被看作是一种在不同的数据库、知识本体开发工具或应用系统之间传递知识本体的中介语言。基于 XML 的语法使得 XOL 可以在一个平面文件 (flat file) 中描述知识本体，并容易通过 web 方式在不同应用开发者中间传递。XOL 为人可读的，在适当复杂度的情况下可以被程序解析。XOL 是一种受限比较严格的语言，仅能对概念、分类以及二元关系进行表述，并且 XOL 没有提供推理机制。XOL 没有专门的编辑工具，但可用 XML 编辑器生成 XOL 文档。

OML 与 CKML

OML⁴ (Ontology Markup Language) 是 1999 年由美国华盛顿大学 (Washington University) 开发的一种对知识本体进行说明的语言。OML 建立在描述逻辑和概念图 (conceptual graphs) 的基础上，允许用艺术逻辑语言来表示概念、分类、关系和公理。OML 由四层组成，OML Core: 最基本，用该层来进行比较; Simple OML, 该层负责直接与 RDF/RDFS 映射; Abbreviated OML, 该层包括概念图特征 (features); Standard OML, 该层是最完备的一级。OML 没有专门的编辑工具，可以 XML 编辑器代替

CKML (Conceptual Knowledge Markup Language)⁵ 可以看作是建立在 OML 之上并对其进行扩展的知识本体描述语言。CKML 和 OML 一样同时具有描述逻辑和框架的特征，都采用 XML 来表示语法。CKML、OML 和 simple OML 的关系如图表 2 所示。



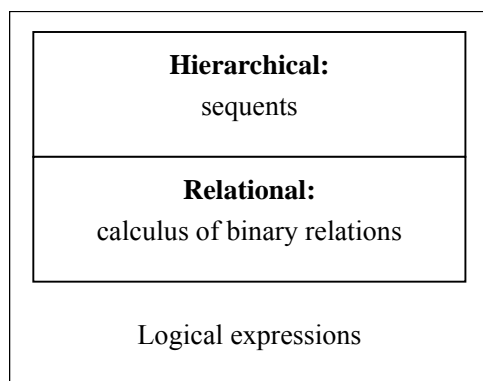
⁴ <http://www.ontologos.org/>

⁵ www.infoloom.com/gcaconfs/WEB/ts1313/tp1313.HTM

图表 1 CKML、OML、SIMPLE OML 的关系

Simple OML 也成为核心 OML。OML 主要特征集中于一个基础的“分类映射图”(classification projection diagram)上,而 simple OML 的主要目标是表达这个图的语义, simple OML 还定义了其与 RDF (S)和 XML Schema 之间的交互性。除了作为 CMKL 和 OML 的核心以外, simple OML 能表示函数、具体化(reification)、基数约束、逆反关系及集合。

OML 用来表达知识本体和模式的结构。知识本体的结构包含类、关系、对象及约束。OML 用图表 3 的三层约束表达(参见图表 3),自上而下分别是序列(sequents)层、二元关系算子层和逻辑表达层。序列(sequents)层对信息流的理论约束(theory constraints)进行建模。二元关系算子层的设计主要考虑针对二元关系约束在实践上的重要性以及处于核心地位的分类投射语义相一致。而逻辑表达层采用声明的方式来表示概念图的知识模型,从而使其与概念保持严格一致。



图表 2 OML 的三层约束表达

CKML 为分布的信息提供了一个概念化的知识表达框架。在 OML 的元素基础上, CKML 还包含了信息流(information flow)的基本元素(分类、理论、解释及局部逻辑)。因此 CKML 实在概念图、正规概念解析及信息流的基础上建立起来的非常接近于一种基于知识本体建模的描述逻辑语言。

RDF 简介

语义 web 的首要目的就是要让计算机能够对信息的语义进行处理, W3C 标资源描述框架(Resource Description Framework, RDF)为基于元数据的语义表示提供了基础。RDF 为在 web 上应用系统间进行机器可理解信息的交换提供了互操作能力。为了描述机器可处理的数据的语义, RDF 定义了一个基本的数据模型,其包含三种对象类型:

- 资源(resources): 一个资源可以是一个完整或部分的网页、网页集合、不需通过 web 访问的任意对象。通常资源用 URI 来命名。
- 属性(properties): 属性使用来描述资源的一个特定方面、特征、品质及关系等。
- 声明(statements): 一个 RDF 的声明是一个特定资源和一个被命名的属性加上这个属性的取值形成的集合。

一个声明由三个部分组成: 主语(subject)、谓语(predicate)、宾语(object)、从其核心来看, RDF 定义了一个“对象-属性-取值”三元组作为

其基本的建模原语并在其之上引入了一套标准的语法。例如：

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org">
    <publisher> world wide web consortium</publisher>
  </rdf:Description>
</rdf:RDF>
```

表示 <http://www.w3.org> (主语) 的 publisher 是 (谓语) W3C (宾语)。既然许多声明的主语和宾语都可以是资源，那么许多声明就可以连成链：

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/home/lassila">
    <Creator
rdf:resource="http://www.w3.org/staffId/857140">
  </rdf:Description>
  <rdf:Description about="http://www.w3.org/staffId/85740">
    <Email>lassila@w3.org</v:Email>
  </rdf:Description>
</rdf:RDF>
```

说明 “<http://www.w3.org/home/lassila>” (主语) 是由编号为 85740 的职员 (宾语) 所创建的，在接下来的声明中，相同的资源 (编号为 85740 的职员) 扮演了主语的角色，并声明了他的电子邮件为 lassila@w3.org。最后，RDF 声明本身也是资源，所以声明可以反复在被用于声明，从而允许形成网络。

RDF Schema 简介

RDF 所提供的建模原语非常基础，只是提供了一个模型，因此需要对其作进一步扩展。RDF Schema 在 RDF 基础上增加了许多语义原语，用来更进一步增加对资源语义上的描述能力，如类、属性、类和属性之间的隶属关系等。常用的 RDF Schema 原语包括：rdf:Resource、rdfs:Class、rdfs:Liternal、rdf:Property、rdfs:range、rdfs:domain、rdf:type、rdfs:subClassOf、rdfs:subPrppertyOf 等。这些描述机制是单纯的 RDF 所不具备的。

另外对于 RDF Schema 和 XML Schema，除了名字上的相似以外，并没有角色上的相同之处。XML Schema (包括 DTD) 描述的是一个 XML 文档中所使用的标签 (tag) 的顺序和组合，定义了 XML 的语法；而 RDF Schema 提供的是对 RDF 建模表示的声明进行解释说明的信息 (语义)，但并不对一个 RDF 描述的语法外观进行约束。RDFS 虽然能表示语义，在某一程度上也能用它来表示 ontology，但是它没有足够的 vocabulary (可以理解为标签) 来表示完整意义上的知识本体。

RDF /RDFS 的工具具有：Amaya, Protégé, Mozilla, SilRI 等等。

我们将在第二节介绍 OWL 语言时说明相关的 RDF、RDFS 原语的使用方法。

Ontology Inference Layer (OIL)

作为一种语义 web 语言的 OIL⁶ (Ontology Interface Layer) 是 On-To-Knowledge 计划的产物。On-To-Knowledge 计划时欧洲的几所大学、研究机构公司于 1999 年发起的，其目标是支持有效的基于知识的管理，注重于对网上弱结构化 (weakly-structured) 信息资源的知识获取、表示及访问。OIL

⁶ www.ontoknowledge.org/oil/

的指导委员会则是由美国和欧洲的几所大学和研究机构共同组成。OIL 的实现基础来自于三个方面：描述逻辑，提供正规语义和推理支持；基于框架的系统（Frame-based），提供认识论上的建模原语；基于 XML 和 RDF 语法的 web 标准。

OIL 是在 RDFS 基础上建立起来的，其对 RDFS 的语义表示能力又作了进一步的扩展，这样使得 OIL 能够对 RDFS 所不能表达的语义进行表达。另外，OIL 被设计为完全兼容 RDF (S) 标准，OIL 文档本身也是一个合法的 RDF (S) 文档。能很好的表示 ontology，并且能最大限度的与 RDFS 兼容，即可以相互转换。

OIL 的设计目标如下：

1. 提供描述基于框架和面向描述逻辑的知识本体所使用的大多数通用的建模原语。
2. 具有简单、清晰和定义良好的一阶逻辑语义
3. 提供自动的推理支持，又曼彻斯特大学开发的 FaCT 系统及 DL (Description Logic) 推理器来完成。

OIL 分四个层次，Core OIL 层，该层实现直接与 RDF/RDFS 映射；Standard OIL 层:Instance OIL 层，该层允许 Concepts 有实例；Heavy OIL 层，该层是最完备的一级。

OIL 的编辑工具有：OILED, Protege2000, WebODE。

DAML+OIL

2000 年 8 月，美国 DARPA 启动了一个为期六年的计划，目的是发展一系列技术使软件 Agent 能够对信息资源进行动态地确认和理解，并为 Agent 之间提供基于语义上的互操作能力。DAML (DARPA Agent Markup language) 是这个计划第一阶段所创建的一种语义 web 语言，它允许用户在其数据上标记语义信息，从而使计算机能对所标注的信息资源进行“理解”。

2000 年 12 月，美国和欧洲两个组织成立联合委员会将 DAML 和 OIL 合并，命名为 DAML+OIL⁷，并提交给 W3C 讨论，使其成为未来语义 Web 标准描述语言的基础。DAML+OIL 也是在 W3C 早期的标准如 RDF 和 RDF Schema 基础上建立起来的，并且用丰富的建模原语对它们进行了扩展。

DAML+OIL 知识基础是 RDF 三元组的集合。DAML+OIL 使用自己的词汇给 RDF 三元组以具体的意思表述。DAML+OIL 将整个世界划分为两个不相交的部分。一部分是由属于 XML Schema 数据类型 (datatype) 的值所组成的，称作数据类型域。另一部分则是由 (单个) 对象所组成的，这些对象应被看作是 DAML+OIL (或 RDF) 中所定义的类的成员，此部分称作对象域。

DAML+OIL 的编辑工具有：OntoEdit, Protege2000, WebODE。

OWL⁸

DAML+OIL 在提交给 W3C 后，发展成了 OWL (Web Ontology Language)。OWL 作为 W3C 的推荐标准⁹，是其所倡导的语义万维网 (Semantic Web) 的核心技术之一，意在提供一种语言，能够用于描述 Web 文档和应用中固有的类和类之间的关系。它通过定义类和类的属性来形式化一个领域，声明和定义对象和对象的属性，以及在 OWL 形式化语义允许程度上对类和对象进行推理。

⁷ <http://www.w3.org/TR/daml+oil-reference>

⁸ <http://www.w3.org/TR/2004/REC-owl-guide-20040210/>

⁹ <http://www.w3.org/TR/2004/REC-owl-features-20040210/>

OWL 建立在 RDF 和 RDF Schema 的基础上，但增加了更多的词汇，具有更强大的描述能力来描述类之间的关系（如：“剥离（disjointness）”），集的基数（cardinality）（如“恰好是 1”），等同关系，更丰富的属性类型和属性特征（如“对称（symmetry）”）等等。

OWL 语言提供三种表达能力依次增强的子语言，设计子语言的目的是用于具体的实施团体和用户团体。

- OWL Lite 支持的用户是那些需要一个分类体系和简单约束功能的人。例如当 OWL Lite 支持集的约束时，它只允许集的值 0 或者 1。为 OWL Lite 提供叙词表和分类法的快速移植支持功能，应该比为其它的表达能力更强的子语言提供这样的支持功能更为简单。
- OWL DL 支持的用户是想获得最大表达能力，完全计算能力（所有的推论都可计算）以及确定性（所有的计算都在限定时间之内完成）。OWL DL 包括所有 OWL 语言的约束如类型区分（type separation）。（一个类不能同时也是一个对象或者属性，一个属性不能同时也是一个对象或者类）。OWL DL 之所以这样命名是由于它和描述逻辑（Description Logics，一个研究一阶逻辑的某一部分的研究领域）¹⁰的一致性（correspondence）。OWL DL 的设计目的是支持现有的描述逻辑和为推理系统提供预期的计算属性（computational properties）。
- OWL Full 支持的用户是想获得最大的表达能力但不确定是否需要计算性，并的 RDF 句法上的自由的用户。例如，在 OWL Full 中，一个类能同时作为对象的集合，它本身也可以作为一个对象。与 OWL DL 的另一个很大的不同是：一个 owl:DatatypeProperty 能标记为：owl:InverseFunctionalProperty。OWL Full 允许一个本体增加一个前控（RDF or OWL）词表的意义。需要注意的是，任何推理软件都不可能支持 OWL Full 的每个功能。

每个子语言都是比它简单的前一个子语言的扩展，不仅扩展了能被合法地表达的事物，还扩展了能被有效地推理的事物。下面的关系是正确的，反之则不正确。

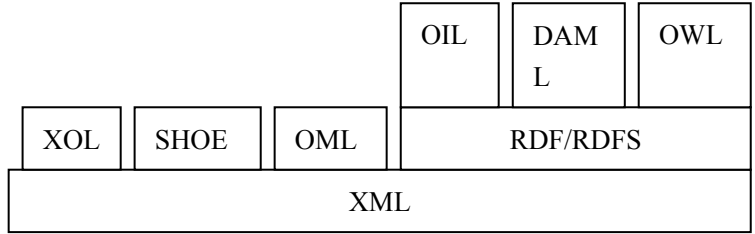
- 每个合法的 OWL Lite 本体是一个合法的 OWL DL 本体。
- 每个合法的 OWL DL 本体是一个合法的 OWL Full 本体。
- 每个有效的 OWL Lite 推论是一个有效的 OWL DL 推论
- 每个有效的 OWL DL 推论是一个有效的 OWL Full 推论。

采用 OWL 的本体开发者应该考虑到哪种子语言更适合他们的需要。选择 OWL Lite 还是 OWL DL，要根据用户对 OWL 所提供的表达能力的约束能力的需求程度。用于 OWL Lite 的推理器（Reasoners）将有令人满意的计算属性，而用于 OWL DL 的推理器在遇到其它具有确定性的子语言时，将按最坏情况处理，服从复杂性更高的子语言。选择 OWL DL 还是 OWL Full 主要根据用户对

¹⁰ <http://www.w3.org/TR/2004/REC-owl-guide-20040210/#DescriptionLogics>

RDF Schema 建模工具的需求程度（例如：定义类的类）。OWL Full 和 OWL DL 比较，对推理的支持比预期的相对较弱。

各种知识本体语言及其与 XML、RDF 的关系



图表 3 各种知识本体语言及其与 XML、RDF 的关系图

知识本体语言还有许多欠缺，离真正的系统实用还有一段距离要走；OWL 有 W3C 支持，又吸取了其他知识本体语言的精华，有望成为最流行的知识本体描述语言，尤其在 web service 与语义网络方面。

知识本体的创建

知识本体有多种形式。元数据方案本身可以看成是知识本体的一种形式，或者一类简单的本体。元数据方案的制定是对一个应用系统相关的实体进行分析并提取属性的过程，如果在此基础上继续对所涉及的各种实体类型的关联关系进行详细分析，最终就能导出本体模型。元数据方案的制定过程就是一种知识本体的建立过程。

除此之外，许多叙词表和分类表等本身就是领域知识的概念体系，包含丰富的关系，虽然许多关系可能不是非常严密，但它们也都可以看成是一类知识本体。上述两种知识本体的初级形式经过规范化和形式化之后，都可能成为计算机可以操作的本体工具。

目前支持本体开发的工具多达数十种，功能各不相同，对于本体语言的支持能力、表达能力、逻辑支持能力以及可扩展性、灵活性、易用性等都相差很大，其中较著名的有 Protégé-2000、OntoEdit、OilEd、Ontolingua 等。Protégé-2000 是目前较活跃的本体工具，是可以免费获得的开放软件，目前的版本是 2.0.1 版，已经有 16500 多注册用户使用。它用 Java 语言开发，通过各类插件支持多种本体格式，甚至已经能够支持刚刚发布的，也是目前最有前途的 W3C 的 OWL 格式。

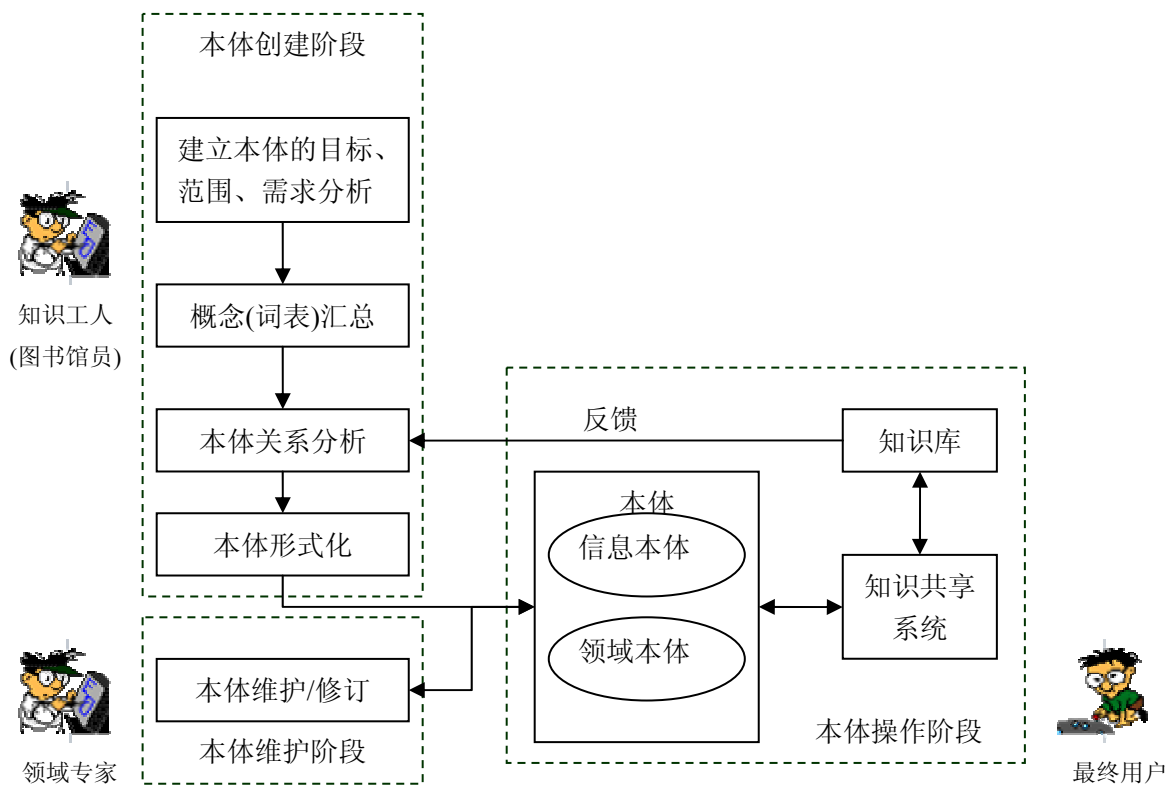


图 3: 知识本体的建立和应用

下面分别介绍一些主要的本体创建的工具，并对这些工具进行简要的比较。

Apollo¹¹

Apollo 是一种友好的知识本体开发应用。图表 6 是 Apollo 的界面，界面的左上部分是知识本体的列表，左下部分是类和实例的列表。当选择了一个类或实例，该类或实例的详细信息将显示屏幕的右半部分。类或实例的槽和值就可以用电子表的形式添加。

¹¹ <http://apollo.open.ac.uk/index.html>

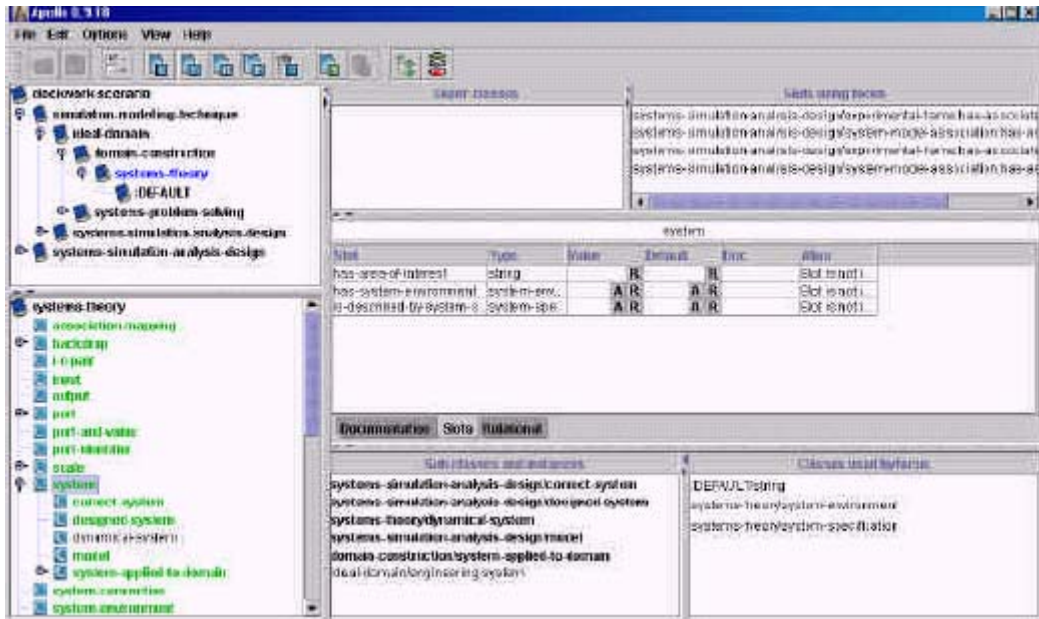


图 4: Apollo 的界面

Apollo 支持知识模型中的所有原语：知识本体、类、实例、功能和关系。在编辑过程中完成一致性的检测。Apollo 有自己内部的存储知识本体的语言，也可以把知识本体输出到其他描述语言中，这根据用户的要求来定。Apollo 采用 Java 语言。

LinkFactory®¹²

LinkFactory®是由 Language & Computing nv 开发的一个形式化知识本体管理系统，用来建设和管理非常庞大和复杂的独立于语言的形式化知识本体。LinkFactory® 由两个主要的组件构成：LinKFactory®server 和 workbench (客户端组件)，两个组件都由 JAVA 开发。

在服务器端，LinkFactory®把数据存储于关系数据库中。数据库的访问被抽象成处理知识本体的一个功能集：获取儿子、发现路径、合并概念、从概念中获取术语等。软件客户端通过标准的 API 访问这些功能，在对数据库内部结构不了解的情况下，这些 API 实现在语义数据库顶层的应用。这个组件能够处理多个并行用户，独立于平台。应用要求运行 RMI 注册（一种 RMI 服务器的域名服务器），从而使客户能够连接到 RMI 服务器。

Workbench 组件允许用户对多个知识本体浏览和建模和结盟。workbench 通过 JAVA beans 实现的一个动态框架。每一个 beans 有特定的功能，有基本的形式知识本体的受限的试图，把 beans 的一个集合和并可一位用户提供浏览和管理数据的有效工具。Java beans 的例子有：概念树、概念准则、概念的详细定义、链接类型树、规则列表、术语列表、搜索格 (pane)、属性格 (pane)、反向关系等。用户可以利用这些 beans 生成多个视图，这些视图称为布局。每一个布局由多个框架构成，这些框架是为了放 beans 用的。生成一个新的布局和在布局中添加一个框架非常简单，可以利用菜单来完成。每一个框架可以分割成多个区域，每个区域放置一个 beans。用户可以通过选择提供的 beans 并把它们拖放到欲放置的领域即可。当用户把 beans 放置在布局中后，就可以在 beans 之间生成链接。

Java beans 不仅可以在 LinkFactory Workbench,中实现相互之间的连接而且可以用于工

¹² <http://www.landc.be/>

SHIQ 描述逻辑来检验一致性。这实现了用户描述知识本体类，用推理器来决定定义在概念体系中的恰当位置。图表 8 表示的是概念定义被判断为不合适的情形。

DAML+OIL RDF Schema 用来装载和存储知识本体。除此之外该工具可以以纯 RDF 文件形式读写概念结构，并可以把知识本体定义成用 HTML 浏览的知识本体，也可以把知识本体定义成 SHIQ，为后期 FaCT 推理器进行分类。概念体系结构可以生成 AT&T dotly 工具可读的格式。OILEd3.4 版用 JAVA 语言开发，可以从 OILEd 站点上免费获取。

OntoEdit Free and Professional versions¹⁴

OntoEdit 是一个知识本体工程环境，支持采用图形方式构建和维护知识本体。OntoEdit 建于内部知识本体模型的顶层。在本题工程生命周期的不同阶段有不同的知识本体支持模型的图形视图。该工具允许用户编辑概念和类的层次结构。这些概念可以是抽象的也可以是具体的，这些概念指出是否可以包含实例。一个概念可以有多个名字，这为概念定义了同义词，该工具提供简单的复制、粘贴功能。该工具基于灵活性大的插入式框架，可以实现以组建化方式扩展工具的功能。插入式界面是公开的，用户可以方便的为 OntoEdit 添加功能进行扩展。提供插件集为用户提供了个性化的工具应用，根据不同的用途场景个性化的调整工具。

OntoEdit 的所有版本都有免费和专业版两种。专业版包括额外的插件集，比如合作环境和推理能力。OntoEdit 的专业版相对于免费版而言，还扩展了其他的功能，如一致性检验、分类和规则执行的推理插件；知识本体的合作工程；管理知识本体库、知识本体的合作共享和长久存储的知识本体服务器。

OpenKnoME¹⁵

KnoME 是一套用 GRAIL 概念模型语言来合作开发知识本体的工具。Tigger 是工具中的一个重要组成部分，用来从没有经过知识本体工程培训的领域专家那里迅速获取知识。这些工具是免费的可以在 OpenKnoME 的网站上免费获取。

GRAIL 的特征有：

求精(refinement)，协调传递与包含的关系。

约束 (Sanctioning)，描述范畴与属性结合在一起生成新的定义时的约束条件，并明确可感知的和有意义的条件。

非本征 (Extrinsics)，未定义的知识与概念框架的连接，使默认推理中为特定信息的应用提供索引成为可能。

用分段语法来生成 GRAIL 的自然语言表示。

KnoME 不是一个独立的系统，它可以与 GALEN 术语服务器 (TeS)通过良好定义的 API 进行通信。GRAIL 资源北转化到经过编译的概念模型。TeS 利用不同的模块提供不同的服务来存储和维护概念模型。API 使知识本体和使用知识本体的客户之间有明显的区别。知识本体被看作一个服务而不是数据结构。通过服务，KnoME 可以对知识本体进行浏览、探索、观察和质量控制。因为知识本体是一个服务，知识本体的传递不是把它输出到一个静态的表单中而是作为一个 TeS 被客户请求和使用。

Protégé-2000¹⁶

¹⁴ http://www.ontoprise.de/com/start_downlo.htm

¹⁵ <http://www.topthing.com>

¹⁶ <http://protege.stanford.edu>

Protégé-2000 由斯坦福大学为知识获取而开发的一个工具。Protégé-2000 可以免费下载，它提供了一个图形和交互式的知识本体设计和基于知识的开发环境。协助知识工程师和领域专家完成知识管理任务。知识本体开发人员可以在需要时迅速访问相关的信息，可以直接实施导航和管理知识本体的操作。树型控制实现了在类层次结构中进行迅速和简单的导航。Protégé 采用表单作为输入槽值的界面。

Protégé-2000 的知识模型与 OKBC 兼容。包括支持类和类层次结构的多继承，模板和私有槽，槽的任意面和定义前的明确说明，明确说明包括值、基数约束、默认值、逆转槽、元类和元类的层次结构。

除了高度方便使用的界面，Protégé-2000 有两个重要的特征，使它从多数的知识本体编辑环境中脱颖而出，即可伸缩性和可扩展性。开发者可以用 Protégé-2000 来构建和使用包括 150,000 个框架的知识本体。对包括成千上万个框架的知识库的支持包括两个组件，一个是后端的数据库来对数据进行存储和查询，另一个是缓冲机制，解决的问题是一旦框架的个数超出了内存的限制，如何加载一个框架。

Protégé-2000 体系结构的最主要的优势就是它的开放的模块化的风格。基于组件的体系结构使系统开发者可以通过生成恰当的插件了增加新的功能。插件可以分为三类，一类是后端插件，使用户可以以多种格式来存储和输入知识库；一类是 slot widgets 类插件，用来为特定的域或特定的任务合并槽或显示和边界槽值；第三类是 tab 插件，通常与 Protégé 知识库一切，提供基于知识的应用。后端插件支持在 RDF Schema、带 DTD 的 XML 文件、XML Schema 文件中存储和导入知识本体。slot widgets 插件包括显示 GIF 图片和音频视频的用户界面组件。tab 插件非常普及，提供高级可视化、知识本体合并、版本管理、推理等功能。例如 tab 插件中的 OntoViz 和 Jambalaya 提供知识库的不同视图，Jambalaya tab 允许交互式的导航、对结构中的特定的元素缩放、用图像中节点的不同层次来强调数据群集之间的连接。

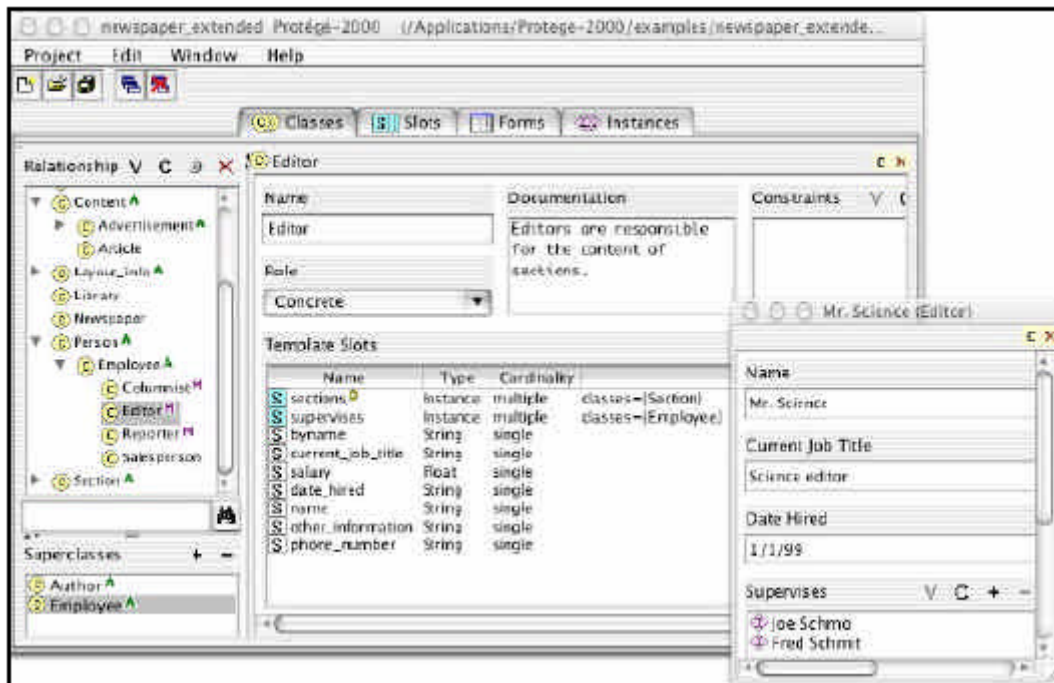


图 6：一个 Protégé-2000 编辑类和槽并输入实例信息的界面

图表 8 中左半部分是类的层次结构和类之间的继承关系。用户可以拖拉类来重新组织类

的层次结构，右半部分显示被选的类的详细信息，包括类的实例的槽的描述。第二个窗口显示的是编辑实例的表单。

PAL tab 提供对 Protégé 公里语言的支持，PAL 是 KIF 的一个子集，当数据的框架形式化并不充分时，用户可以对他们的数据进行限制。PAL 推理引擎就对这些限制的数据进行分析，告诉用户知识库中的哪些限制没有被遵守，情况如何。Flora tab 和 Jess tab 提供在其他地方访问开发的推理引擎。PROMPT tab 提供管理多个知识本体的环境，它的组件包括知识本体合并工具，帮助用户发现原知识本体之间的相似之处，并对知识本体进行合并。UMLS 和 WordNet tabs 使用户可以把大的在线知识资源的元素导入和集成到自己的知识本体中去。

对上述知识本体工具进行比较的情况如下：

这些工具的一般描述见图表 9。

工具	开发机构	最新版本	可用性
Apollo	KMI(Open University)	1.0 Beta 3 (MAY 2002)	开放资源
LinkFactory	Language & Computing nv	May 2002	网站许可证
OILEd	University of Manchester	3.4 (Apr2002)	开放资源
OntoEditFree	Ontoprise	2.5 (May2002) 3.0 (Aug2002)	免费软件
OntoEdit professional	Ontoprise	2.5 (May2002) 3.0 (Aug2002)	需要软件许可证
Ontolingua	KSL (Stanford University)	1.0.649 (Nov2001)	Web 免费访问
Ontosarus	ISI (University of Southern California)	1.9 (Mar2002)	开放资源和 web 免费访问
OpenKnoME	University of Manchester	5.4	免费软件
Protégé 2000	SMI (Stanford University)	2.0.1 (Feb2004)	开放资源
SymOntoX	LEKS (IASI-CNR)	1.0	Web 免费访问
WebODE	Ontology Group (UPM)	2.0 (Mar2002)	软件许可证和 web 免费访问
WebOnto	KMI (Open University)	2.3 (May2001)	Web 免费访问

表 1：工具的一般描述

比较的方面主要有利用工具的软件平台和硬件、体系结构（单一结构或 C/S 结构或 n 层结构）、扩展性、知识本体的存储和备份管理。从发展前景看，多数的工具正朝着 JAVA 平台发展，结构一般具有扩展性。知识本体工具的一个弱项是在数据库中的存储。只有 LinkFactory, OntoEdit Professional Version, Protégé2000 and WebODE 使用数据库来存储知识本体。对于备份管理功能只有 OpenKnoME, SymOntoX, WebODE and WebOnto 提供数据库备份。图表 10 是知识本体体系结构的比较。

工具	Sw 体系结构	可扩展性	知识本体存储	备份管理
Apollo	单一结构	插件	文件方式	否
LinkFactory	3 层结构	可以扩展	数据库关系系统存储	否
OILEd	单一结构	否	文件方式	否
OntoEditFree	单一结构	插件	文件方式	否
OntoEdit proessional	单一结构 &C/S 结构	插件	文件和数据库方式	否
Ontolingua	C/S 结构	无	文件	否
Ontosarus	C/S 结构	无	文件	否
OpenKnoME	C/S 结构	无	文件	有 (audit 日志)
Protégé 2000	单一结构	插件	文件和数据库管理系统	无
SymOntoX	3 层结构	无	XML	是
WebODE	3 层结构	插件	数据库管理系统	是
WebOnto	C/S 结构	无	文件	是

表 2 : 知识本体工具的体系结构

互操作性是知识本体工具的另一个重要指标,指知识本体开发工具与其他知识本体开发工具、知识本体合并工具、信息系统和数据库之间的互操作性,也指从一种知识本体语言到另一种知识本体语言的相互翻译,互操作性在知识本体应用总集成知识本体中起到重要作用。许多知识本体工具可以导入或到处临时的 XML 文件或其他标记语言。图表 11 是知识本体工具互操作的比较。

工具	与其它知识本体工具的互操作	从其他语言导入	导出到其他语言
Apollo	无	Apollo metalanguage	OCML、CLOS
LinkFactory	FastCode TeSSI	XML、RDF(S)、 DAML+OIL	XML、RDF(S)、DAML+OIL、 HTML
OILEd	FaCT	RDF(S)、OIL、 DAML+OIL	OIL、RDF(S)、DAML+OIL、 SHIQ、dotty、HTML
OntoEditFree	OntoAnnotate、 Ontobroker、 OntoMat、 Semantic、Miner	XML、RDF(S)、 Flogic、 DAML+OIL	XML、RDF(S)、Flogic、 DAML+OIL
OntoEdit proessional	OntoAnnotate、 Ontobroker、 OntoMat、 Semantic、Miner	XML、RDF(S)、 Flogic、 DAML+OIL	XML、RDF(S)、Flogic、 DAML+OIL、SQL-3
Ontolingua	Chimaera、CML Model、 Fragment Editor、 Equation Solver、 Data structures inspector	Ontolingua、IDL、 KIF	KIF 3.0、CLIPS、CLIPS sentential、format、CML ATP、CML rule engine、 EpiKit、IDL、KSL rule engine、LOOM、OKBC

	Expressions Evaluator、OKBC		syntax、PROLOG syntax
Ontosarus	--	LOOM、IDL、ONTO、KIF、C++	LOOM、IDL、ONTO、KIF、C++
OpenKnoME	GCE(GALEN CASE Environment)、SPET (Surgical Procedure Entry Tool)	GRAIL、GALEN IR	GRAIL、CLIPS、HTML、GALEN IR
Protégé 2000	PROMPT、OKBC、JESS、FaCT	XML、RDF(S)、XML Schema	XML、RDF(S)、XML、Schema、Flogic、CLIPS、Java、HTML
SymOntoX	--	--	RDF(S)
WebODE	JESS、PICSEL、OILEd、ODEMerge、ODE-KM	XML、RDF(S)、CARIN	XML、RDF(S)、OIL、DAML+OIL、CARIN、Flogic、Prolog、Jess、Java、HTML
WebOnto	PlanetOnto、ScholOnto、MnM	OCML	OCML、Ontolingua、GXL、RDF(S)、OIL

表 3: 知识本体工具互操作比较

从知识表示的角度来看, 知识本体工具可以分为两大类, 一类是如 OILEd、OntoSaurus 和 OpenKnoMe 的基于描述逻辑的工具; 另一类是遵循基于框架和一阶逻辑的混合方法来表示知识。此外, Protégé2000 提供象元类 (metaclasses) 的模型组件。关于知识本体工具支持的方法, OntoEdit 的两个版本均支持 OntoKnowledge 方法。OpenKnoMe 工具支持 GALEN 方法, SymOntoX 支持 OPAL 方法, WebODE 支持 Methontology。然而, 所比较和分析的工具没有一个包括项目管理功能和知识本体维护功能, 这些工具只是提供了简单的知识本体评价支持。

选择一个工具前, 还要考虑的一个因素是工具是否附加推理服务。自动推理服务包括内置的推理引擎和引进的推理引擎、约束和一致性检验机制、自动分类和异常处理等。LinkFactory 有自己的推理引擎, OILEd 采用 FACT 推理引擎实施推理, OntoEdit 专业版采用 OntoBroker, Ontolingua 采用 ATP, Ontosaurus 采用 Loom 分类器, OpenKnoMe 拥有自己的推理引擎, Protégé-2000 采用 PAL, SymOntoX 拥有自己的推理引擎, WebODE 采用 Ciao Prolog, WebOnto 采用 OCML 推理引擎。除此之外, WebODE 和 Ontosaurus 提供评价功能。只有 LinkFactory、OILEd、OntoSaurus 和 OpenKnoMe 提供自动分类服务。没有一个工具提供异常处理机制。

关于工具的可用性, WebOnto 具有最先进的合作构建知识本体的功能。一般来说, 工具需要包含更多的促进成功合作构建知识本体的功能。另外与帮助系统、编辑和可视化等方面有关的其他的可用性应该在知识本体工具开发中得到足够的重视。图表 12, 是关于知识本体工具可用性方面的比较。

工具	图形分类系统	视图的图形修剪 (prunes)	缩放	合作working	知识本体库
Apollo	是	是	否	否	是

LinkFactory	是	是	是	是	是
OILEd	否	否	否	否	是
OntoEditFree	否	否	否	否	否
OntoEdit professional	否	否	否	是	是
Ontolingua	是	否	否	是	是
Ontosarus	否	否	否	是	否
OpenKnoME	否	否	否	是	是
Protégé 2000	是	是	是	否	是
SymOntoX	是	是	否	是（合作系统）	是
WebODE	是	是	否	是	否
WebOnto	是	是	否	是	是

表 4：知识本体工具可用性比较

从上面的比较可以看出，构建知识本体的开发工具有许多的相似之处，但这些工具既不互操作也不能涵盖知识本体开发生命周期中的所有活动。当要把知识本体集成到由不同知识本体工具组成的知识本体库中时，知识本体工具之间缺乏互操作将会引起严重的问题，这种情况也会发生在用知识本体合并工具来合并由不同的工具和语言构建的两个知识本体时。

因此需要生成一个知识本体开发者的通用工作台，协助在整个知识本体生命周期中知识本体的开发和构建，协助知识本体内容可视化中知识本体管理的复杂技术。该知识本体开发通用工作台应该有支持其他系统使用知识本体的知识本体中间件服务集合，这些服务包括为特定的应用定位最恰当的知识本体的软件；在相同的术语或不同的知识本体之间比较其语义相似性和语义差距的正式量度；能够实现知识本体的增量的、一致性的、选择性的更新的软件，这些软件被给定的应用和对知识本体库远程的访问所使用；还包括协助知识本体与遗留系统和数据库的集成的软件。

今天，知识本体技术已经被广泛应用到企业和公司中，在语义环境中大量基于知识本体的应用开发将会生成一个“知识本体应用程序开发包”，该包能够实现基于组件的现有和未来应用的迅速开发和集成。

问题讨论

知识本体重用时对知识本体的评价问题

在知识本体的建设之前，我们需要查找是否有现成的知识本体可以被我们重用。一旦我们查找到已经存在关于某一个领域的现存知识本体，并下载下来。如果这样的知识本体不仅一个而是多个，我们应该如何对这些知识本体评价，如何确定我们是重用这些知识本体还是自己从头建设一个知识本体。

现有的知识本体评价工具只有 ONE-T，建于 1997 年，所以对于今天的知识本体的评价并不实用。笔者认为对知识本体的评价应该考虑的因素有知识本体的来源、知识本体的使用评价、相同功能的知识本体的应用比较等。其中知识本体的来源是比较重要的因素，不仅包括知识本体来自什么地方，还应该考虑到知识本体的创建者，创建者维护知识本体的程序，知识本体的最初建来为谁服务等。例如在农业领域，来自联合国粮农组织的知识本体就比来

自 wordNet 的知识本体可靠的多。

知识本体的互操作问题

不同的领域有不同的知识知识本体，相同的领域也存在多个组织的不同知识本体，即使是对现有知识本体重用或扩展也会存在用不同的方法给相同的概念建模的情况。因此知识本体之间的互操作是在知识本体建设中必须考虑的问题。

关于知识本体的互操作，有不同的途径，一种是直接翻译，即每一个知识本体都包含一定的翻译功能，通过源知识本体和目标知识本体之间术语的同义影射和关系建立来完成与其他知识本体的影射。这种方式只有在知识本体的数量较少时才可行。知识本体互操作的第二种方式是单一共享知识本体，存在一个复杂的知识本体，所有其他的知识本体都通过复杂知识本体实现互操作。第三种方法是多共享知识本体，通过多个相对简单的知识本体来定位共享的知识。多共享知识本体互操作方法是一种更灵活，更具有伸缩性的方法。不用所有的知识本体都必须遵循某一个非常复杂的知识本体。

所有的知识本体互操作途径，在底层都是通过不同的知识本体语言来实现，因此这些知识本体语言需要能够实现不同的知识本体之间数据的转换，不同的知识本体之间的关联，允许数据在不同的知识本体之间进行转换，形成知识本体网络。虽然共享知识本体和知识本体扩展允许不同的组织和领域之间有某种程度的互操作，但用不同的方法进行信息建模的情况还是经常发生。这要归结于不同的组织、专业，国家的从不同的视点来看待一个问题。为了使机器能整合遵从不同知识本体的信息，需要在开始的时候就把概念映射到其它知识本体中与该概念等同的概念。W3C 的 owl 知识本体语言支持在所有用例中，不同的数据提供者使用不同的术语。并得到了 RDF (s) 的支持。当然 owl 知识本体语言并不是唯一可以采用的语言，本文第四部分已经对知识本体的其他语言进行了描述和介绍，在知识本体建设及互操作中可以对需要采用的语言进行借鉴。

知识本体的使用问题

通过上面的讨论，我们可以看出作知识本体可能并不难，我们可以通过对领域中知识和概念的分析，确定他们之间的关系和可能的约束与规则，选择一定的知识本体构建方法与知识本体建设工具，形成某一个领域中的知识本体。到此为止，工作并不是非常难，难的是如何构建合理有效的知识本体，并把这些知识本体应用到具体的应用领域中，即如何用的问题。站得高点说，就是知识工程和知识本体工程方面的问题；具体点说，就是知识本体的构建和应用问题。该问题，只能通过不断的实践和总结才能够逐步解决。

与元数据标准规范一样，本体的价值在于共享和重用，而且是基于机器之间进行语义理解的共享和重用，因此形式化是非常重要的。形式化语言经过数年的发展目前已基本定位到了 W3C 主导的 OWL 语言，因此对于图书情报界来说，我们目前应该更多地关注图书情报学长期积累的许多“知识体系”，是否能够应用新的模型、采用新的方法进行新的分析，经过规范化之后转化为计算机可以理解的本体形式，应用到数字图书馆建设中去。

参考文献

1. <http://csdl.computer.org/dl/mags/ex/2001/02/x2030.pdf> (2004/04/23)
2. C. Lagoze, "Keeping Dublin Core Simple: Cross Domain Discovery or Resource Description?," D-Lib Magazine, 7 (1), 2001.
3. [Studer, Benjamins, Fensel 1998] Knowledge Engineering: Principles and Methods. Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Data and Knowledge Engineering,

- 25(102):161-197, 1998.
4. D. Brickley, J. Hunter, and C. Lagoze, "ABC: A Logical Model for Metadata Interoperability," Harmony Project, Working Paper 1999. http://www.ilrt.bris.ac.uk/discovery/harmony/docs/abc/abc_draft.html
 5. Cromwell-Kessler, W. (1998). Crosswalks, metadata mapping and interoperability: What does it all mean? In Murtha Baca (ed.), *Introduction to Metadata: Pathways to Digital Information*. Los Angeles: Getty Information Institute. pp.19-22
 6. D. Bearman, G. Rust, S. Weibel, E. Miller, and J. Trant, "A Common Model to Support Interoperable Metadata. Progress report on reconciling metadata requirements from the Dublin Core and INDECS/DOI Communities," *D-Lib Magazine*, 5 (January 1999), 1999.
 7. J. Hunter, "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains," *Journal of Digital Information*, 1 (8), 2001.
 8. J. Hunter and D. James, "Application of an Event-Aware Metadata Model to an Online Oral History Archive," presented at ECDL 2000, Lisbon, 2000.
 9. J. Hunter and C. Lagoze, "Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles," presented at WWW10, Hong Kong, 2001.
 10. ICOM/CIDOC Documentation Standards Group, CIDOC Conceptual Reference Model, 1998 <http://www.ville-ge.ch/musinfo/cidoc/oamodel/>
 11. C. Lagoze, "Business Unusual; How "event awareness" may breathe life into the catalog," presented at Bicentennial Conference on Bibliographic Control in the New Millennium, Library of Congress, Washington DC, 2000.
 12. Sean Bechhofer, Ian Horrocks, Carole Goble, Robert Stevens. OILED: a Reason-able Ontology Editor for the Semantic Web. Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396--408. 2001.
 13. Horrocks, U. Sattler, S. Tobies. Practical reasoning for expressive description logics. 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99) (LNAI, Springer-Verlag, 1999). 161-180.