本体变化管理技术研究综述*)

鲍爱华 姚 莉 刘 芳 张维明
(国防科技大学信息系统与管理学院 长沙 410073)

摘 要 语义Web是一个动态变化的环境,作为其基础的本体也需要及时适应环境的变化而进行改进。本文首先对本体变化和本体变化管理的概念进行了分析,将本体变化管理划分为三个部分:本体变化发现、本体变化处理和本体变化结果传播。以这个划分为基础,分别对本体进化、本体版本机制、多版本推理和不一致推理等与本体变化管理相关的技术进行了具体的阐述,并采用一致的标准对这些研究进行了分类比较。还说明了与本体变化管理相关的研究内容。

关键词 本体变化,本体变化管理,本体进化,本体版本机制,多版本推理,不一致本体推理,本体调试,本体模块化

Survey on the Research of Ontology Change Management Technology

BAO Ai-Hua YAO Li LIU Fang ZHANG Wei-Ming
(School of Information System and Management, NUDT, Changsha 410073)

Abstract The semantic Web is a dynamic changing envrionment, so, the ontology, which is the basis of semantic Web, should evolve to cope with the change of enviroment. This paper analyzed the concept of ontology change and ontology change management, and then devided the ontology change management into three parts, i. e. Ontology Change Discovery, Ontology Change Process and Ontology Change Propagation. Then, the research related to the change management, such as ontology evolution, ontolgoy versioning, were surveyed, and compared using the same criterion. The Research Direction related to ontology change management was also introduced.

Keywords Ontology change, Ontology change management, Ontology evolution, Ontology versioning, Multi-version ontology reasoning, Inconsistent ontology reasoning, Ontology debugging, Ontology modularization

1 引言

随着语义Web的发展,作为其基础的本体得到了深入的应用,以本体作为概念框架的应用也越来越多[1]。目前,针对本体的研究绝大部分集中在本体的建模和使用上。在这些研究中,人们假定本体作为领域概念框架是稳定的,相关的应用也能够基于本体稳定地运行。实际上,在语义Web环境中,领域知识是不断演化的[2],作为其概念架构的本体也需要与领域的改变相适应,进行及时的修正,才能更好地展现其领域知识表达能力。因此,关于本体如何适应环境进行改变逐渐成为一个新的研究热点。

目前,关于本体改变的研究散见于多个研究领域,例如在本体进化的研究中,针对本体在进行修改时如何保证其一致性进行了研究;在本体调试的研究中,针对本体在改变时如何发现其不一致性、如何寻找到不一致根源的问题进行了研究;在本体模块化的研究中,针对本体的模块特性进行了研究,使得本体在改变时能够在最小范围内影响本体的使用,降低本体修改的复杂度;在本体版本控制的研究中,针对如何记录本体变化、如何采用多版本的方式对复杂应用提供一致的访问进行了研究;另外,在不一致本体推理的研究中,针对在无法保证本体一致性的情况下如何为相关应用和推理任务提供一致的服务进行了研究,使得本体在变化过程中同样能够提供有效一致的服务,等等。

上述研究内容分别针对不同的需求对本体改变进行了不同侧重的研究。而本体变化管理就是以本体变化为中心,对本体改变的原因(变化发现)、本体改变的处理方式和本体改变的结果传播进行管理,使得在本体改变的整个生命周期内对外提供一致的本体服务,从而不影响相关应用或依赖本体的有效性。

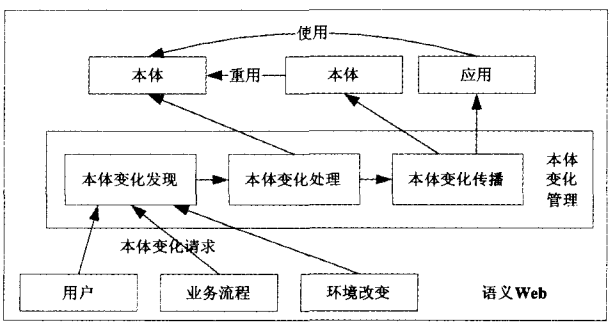

图1 本体变化管理的组成

下面,本文以本体变化管理的三个内容为中心,对各个阶段中目前的研究现状进行分析,从而理清本体变化管理的概念和技术路线,为本体变化管理的进一步研究打下基础。

2 本体变化管理的概念

本体是一个内在关系复杂的知识库,局部修改本体有可

*)本课题得到国家自然科学基金(项目编号 70371008)资助。鲍爱华 博士研究生,主要研究方向为分布式人工智能、语义Web;姚 莉 教授,博导;张维明 教授,博导。

能导致整个本体的不一致，从而影响基于本体应用的有效性。例如，在某个概念发生改变时，与其关联的概念或者实例也可能发生变化。本体变化管理以本体变化为中心，重点对本体变化所带来的不一致性进行处理，并将处理结果予以传播，使得依赖于本体的应用不会受到本体不一致的影响。

本体变化管理由三个部分组成(图 1)，即变化发现、变化处理和变化结果传播。这三个部分分别对应着本体不同的生命周期，所采用的技术和目的也不相同。其中，变化发现管理主要采用一定的技术对本体的不一致进行侦测，发现不一致的根源并找到本体所需要进行的变化；变化处理主要采用本体进化技术，对本体中需要修改的部分进行修改，并维护本体的一致性；变化结果传播则主要处理本体变化在依赖于本体的应用中的传播，使得相关应用能够不受本体变化的影响，能够进行有效的推理。

在本体的整个生命周期中，本体可能在多个阶段发生不一致。在本体的协同开发中，多个知识工程师可能由于本体改进意图潜在或显在的冲突而导致本体的不一致；在本体的使用过程中，环境可能发生相应的改变(如业务流程发生变更等)，使得原先的本体变为不一致；用户在使用过程中可能发现本体建模不合理，需要知识工程师对本体进行相应的改进；另外，本体的表示方式也可能需要发生改变。

在本体变化的处理过程中，除修改本体外，还需要处理本体改变的边际效应，以保证本体的一致性。例如，在删除某个概念时，其所属的子概念应该进行一定的处理，否则会导致这些子概念由于在概念层次中定位而无法存在。在目前的研究中，本体进化技术[3]能够在一定程度上对本体进行维持一致性的修改。

在本体变化完成后，需要将变化结果传播到依赖本体和应用中去。由于知识工程师事先无法得知哪些本体和应用依赖于修改的本体，因此也无法保证这些依赖本体与应用能够及时适应本体所做的修改。所以，需要采用一定的机制来让相关本体和应用及时发现这些变化，并能够自适应地为没有及时改进的应用提供兼容的服务。变化结果传播主要面对的就是这个需求。在目前的研究中，本体版本机制在一定情况下能够满足这个需求。

3 本体变化管理技术研究现状

下面从本体变化的相关概念入手，对目前涉及到本体变化管理的技术进行分类介绍。

3.1 本体变化

定义 1(本体变化[3]) 本体变化 $oc \in OC$ 是本体之间的映射函数，即 $oc: O \rightarrow O$，其中 OC 代表本体变化的集合。

由于本体在变化后可以进行回溯并恢复到变化之前的状态，因此本体变化 oc 是一个双射函数。

改变本体后可能导致本体的不一致，因此需要对本体的不一致进行管理。关于本体一致性的定义，许多研究工作针对研究重点的不同，给出了不同的定义。总体而言，我们称一个本体是一致的，如果它能够满足相应的本体一致性条件。

文[3]和文[4]将本体的一致性划分为三种类型：语法(结构)一致性、语义(逻辑)一致性和用户自定义一致性。语法一致性是指本体所需要满足的语法规则，例如在 KAON 本体中，概念层次不能存在闭环；在 OWL-Lite 和 OWL-DL 本体中，不能采用概念作为属性等。语义一致性是指在语义上本体模型要求是可满足的，例如 OWL 本体{$Student \sqsubseteq Employee$, $Student(Peter)$, $Employee(Peter)$}在逻辑上是不可满足的，因此在语义上不一致。用户自定义一致性是指在语法和语义一致性以外用户对于本体所制定的个性化要求。例如，要求本体的概念层次中，两个概念之间不存在一条以上的通路，即对于本体 O 中的任何一个公理 $C_1 \sqsubseteq C_n$，O 中不存在任何概念，使得 $C_1 \sqsubseteq C_2 \sqsubseteq \cdots, \sqsubseteq C_{n-1} \sqsubseteq C_n$。

本体一致性是本体变化管理的基础。本体由于改变而违反了一致性原则，因此需要对一致性进行修复，或者基于不一致的本体对相关应用提供一致的服务。

3.2 本体变化发现技术

随着本体规模的扩大，对于知识工程师而言，如何发现本体需要修改也是一个复杂的工程。一般而言，本体的变化来源于两个方面：一是本体构建不符合用户需求，二是环境的变化使得本体变得不一致。针对第一种来源，文[5]提出了一种用户驱动的本体变化发现方法，通过归纳，总结用户对本体的使用情况来获取用户所期望的本体模式，从而对本体进行改进。例如，在以本体为基础的电子商务网站分类体系中，如果用户点击某个二级分类的频率远远超过某个一级分类，就说明该本体的概念分类存在一定的问题，需要进行修正，以满足用户的需求。

环境的变化同样会导致本体的不一致。例如，在描述组织结构的本体中，如果某个组织被裁撤，那么本体中对该组织的描述就变得没有意义，应该予以改进。在对本体进行分析时，可以通过推理技术来分析本体在逻辑上是否一致，例如基于场景的演算方法能够发现 OWL 本体中某个概念是否是可满足的。

尽管通过推理能够判断某个本体是否是可满足的，但是对于大部分知识工程师而言，发现本体不一致的原因仍然是很困难的。通过推理可以发现某个概念是不可满足的，但是由于概念的不可满足性具有可传播特性，因此即使对这个概念进行处理也未必能够达成整个本体的一致性。只有寻找到导致本体不一致的根源，才能快速地找到本体所需要进行的改变。在文[6]中，作者以场景演算为基础，提供白盒技术和黑盒技术两种方式对本体进行调试，除了告知用户哪些概念不可满足，还会给出不可满足的原因，使得用户能够从原因着手，跟踪到导致本体不一致的根源。在文[7]中，作者提出了一种启发式的 OWL 本体调试方法，来帮助用户定位本体不一致的来源。作者首先分析了 OWL 本体中概念不可满足性的传播特性，定义了不可满足性的三种来源：本地不可满足性、传播不可满足性和全局不可满足性，针对三种不可满足性分别制定了启发式的调试流程，从而帮助用户进行不一致的快速定位。

本体模块化技术[8,9]从一定程度上也能够帮助用户快速发现本体的不一致。在大规模本体的实际构建中，往往存在着大量的本体重用。这时，由于存在着相对独立的描述领域，因此可以将本体分割为若干个独立的部分。在本体发生不一致时，不一致性在这些独立分块之间的传播可以得到控制，这也减少了本体变化发现的难度，提高了故障定位的准确度。

3.3 本体变化处理技术

本体变化处理是本体变化管理的核心，它不仅需要对本体实施改进，还需要考虑本体改进所带来的边际效应，维护整个本体的一致性。这时，本体进化可以对本体进行维持一致性的改变。

定义 2(本体进化[3]) 本体进化是指本体根据出现的变

化和由这些变化引起的本体一致性问题进行自适应变更。

下面将本体进化分为四个类型来分析本体进化技术的研究现状。

3.3.1 本体进化框架

本体进化框架是本体进化的基础，目前已经有许多研究提出了比较成熟可靠的框架。在文[3]中，作者提出了一个四阶段本体进化核心过程：

(1)本体变化表示。这一阶段主要将本体改进意图进行形式化描述。例如，如果改进需求是将概念 C_1 的子概念 C_2 提升为 C_1 父概念 C_3 的子概念，可以形式化地表示为{$RemoveSubConcept(C_2,C_1)$,$AddSubConcept(C_2,C_3)$}。形式化描述能够更准确地反映知识工程师的改进意图，从而为后续的分析打下基础。

(2)本体变化语义分析。本体变化语义分析主要对本体变化的边际效应进行分析，即在实施本体变化的同时，需要进行哪些额外的变化才能维持本体的一致性。例如，在删除概念时，需要对该概念的实例、子概念进行处理，才能保证本体的一致性。

(3)本体变化传播。在完成本体变化的语义分析后，需要将这些变化传播到依赖于该本体的应用和本体中去。

(4)本体变化实施。在变化传播完成后，需要在原有本体中实施相应的变化，并将新版本发布，使得其他本体和应用能够共享新的本体知识。

在文[10]中，作者使用多版本技术来进行变化侦测，提出了一个五阶段的进化方法。与上述方法相比，该方法在本体变化语义分析后采用版本比较的方法，将一些不必要的本体变化回溯，从而能够更好地反映知识工程师的本体改进意图。例如，在进行本体概念层次转移时，知识工程师并不打算改变相关实例，但进行本体变化语义分析时，有可能为了保持一致性而对实例数据进行了处理。这时，就可以通过变化回溯的方法来恢复针对实例数据的修改。

3.3.2 基于进化策略的本体进化方法

在 Stojanovic 所提出的四阶段本体进化方法[3]中，本体变化的语义分析是最重要也是最困难的部分。在文[3]中，作者对 KAON 本体的进化问题进行了研究，通过预先定义进化策略的方式，使得用户能够预先为每种情况选择应对措施，从而能够让本体进化系统自动作出合理的进化步骤，供用户选择。例如，在删除某个概念时，需要处理其子概念。通常有三种处理策略，即删除子概念、将子概念转移为该概念父概念的子概念以及将子概念转移为根概念的子概念。用户在本体进化前根据个人偏好事先对这三种策略进行排序，使得本体进化系统在进行语义分析时自动根据用户偏好分析出进化步骤，从而完成语义分析工作。通过进化策略，本体进化系统能够分别为每个知识工程师制定出满足个人偏好的进化结果，同时将变化步骤按照偏好进行排序，由用户选择最终执行的变化顺序，实现个性化的本体进化。

由于本体语法变化的类型是有限的，因此能够针对本体语法变化制定对应的进化策略。但是对于语义变化，因为无法预料某个变化会带来何种后果，因此很难实现制定预定措施。因此，基于进化策略的本体进化方法更多地适合于维护本体语法一致性的本体进化，对于要求语义一致性的本体进化(例如 OWL 本体的进化)则难以适用。

3.3.2 基于事务的本体进化方法

在本体协同开发中，多个开发者可能同步对某个本体进行扩展。这时，如果两个知识工程师同时对某个元素(概念、关系等)进行修改，那就有可能造成本体的不一致。这时，人们借鉴数据库中关于字段读写事务操作的方法，提出了基于事务的本体进化方法。

基于事务的本体进化方法的核心是，将针对本体的进化操作按照事务的方式进行管理，避免造成本体的不一致。例如，在某个知识工程师对概念进行编辑操作时，本体进化系统将锁定该概念，所有其他知识工程师针对该概念的操作均被阻塞，直到该操作完成。

以事务操作的方式来对本体变化进行管理，能够有效地防止知识工程师的操作在没有完成时被其他工程师破坏。但是，由于它没有分析本体变化对本体全局所造成的影响，因此并不能很好地解决本体语义一致性维护问题。一般来说，基于事务的本体进化技术在本体协同开发中能够得到较好的应用，在实现上通常也是集成在本体编辑器中，例如 Protégé 就提供了基于事务的本体协同开发支持。

3.3.3 基于推理的本体进化方法

在许多情况下，维护本体语法一致性是不够的。例如，对于 OWL 本体，如果本体在语义上是不一致的，那么就无法基于该本体进行推理。因此，需要进一步维护本体语义的一致性。在文[4]中，作者分析了 OWL 本体的逻辑一致性，提供了两种方法来维护 OWL 本体的语义一致性。

(1)寻找最大一致子本体法。

定义 3(最大一致子本体) 本体 O' 是本体 O 的最大一致子本体，如果 O' 满足：①；$O'\subseteq O$；②O' 是语义一致的；③ $\forall O''\subseteq O$。如果 $O'\subseteq O''$，那么 O' 是不一致的。

寻找最大一致子本体进化方法的基本思想是：在向本体 O 中添加一个公理 α 后，可以新建一个本体 $O'=O\cup\{\alpha\}$ 和 $O''=\varnothing$，并逐步从 O' 中抽取与 $O''\cup\{\alpha\}$ 不一致的公理添加到 O'' 中，直到 O' 中剩余的公理均不与 $O''\cup\{\alpha\}$ 冲突为止。这时，O'/O'' 就是本体 $O\cup\{\alpha\}$ 的最大一致子本体，同时 O' 也就是本体进化所需要达到的本体。

(2)寻找最小不一致子本体法

定义 4(最小不一致子本体) 本体 O' 是本体 O 的最小不一致子本体，如果 O' 满足：①；$O'\subseteq O$；②O' 是不一致的；③ $\forall O''\subseteq O$。如果 $O''\subseteq O'$，那么 O' 是语义一致的。

与前一种方法相反，该方法首先寻找最小不一致子本体，然后根据用户需求从最小不一致子本体中删除任何一个公理均可以得到一个语义一致的本体，达到本体进化的目的。寻找最小不一致子本体的方法与寻找最大一致子本体的思想相反，这里不再赘述。

在上述两种方法中，对公理与本体之间的一致性分析均通过场景演算的推理方式实现，因此称这两种方法为基于推理的本体进化方法。基于推理的本体进化方法可以对 OWL 本体进行维持语义一致性的改进。

3.3.4 **基于信念修正的本体进化方法**

前述的几种本体进化方法虽然能在一定程度上维护本体语法或语义的一致性，但是由于无法脱离人工的干预，使得这些方法在处理大规模本体的进化问题上存在一定的局限性。在文[11]中，作者分析了知识库中信念修正与本体进化的相似性，认为可以借鉴知识库中信念自动更新的优点，使得本体能够脱离人工干预而进行自动的进化。作者认为，如果将本体看作为一个知识库，本体进化与信念修正非常相似。例如，在信念修正中，将新的信念引入时，需要对原有信念进行分

析，将不一致的信念及推理结果进行处理；在本体进化中，除了对本体进行修改外，还需要对变化的边际效应进行处理，以维持本体的一致性。

但是，知识库中的信念修正算法是无法直接在本体进化中使用的。这是因为，在信念修正中，知识库是基于封闭世界假设的，而在语义 Web 环境中，本体(OWL)基于开放世界假设[12]，两种不同的世界假设导致了前者的理论无法直接应用到后者上来。例如，在实例检测问题中，使用知识库进行推理时，如果无法推导出个体 a 是概念 C 的实例，那么就可以说明个体 a 不是概念 C 的实例；而使用 OWL 本体进行推理时，如果无法推导出个体 a 是概念 C 的实例，那么只能说明根据目前的知识无法判断 a 是否是 C 的实例，而无法作出否定的回答。在文[12]中，作者对信念修正中最为基础的 AGM 理论[13]进行了分析，将 AGM 的逻辑基础进行了扩充，使之能够部分适应于基于开放世界假设的本体知识库更新。

与前面的几种方法相比，基于信念修正的本体进化方法能够自动地进行本体维护，可以大量减少知识工程师的工作量，具有很大的优越性。目前，关于信念修正在本体进化上的应用主要处于初步的理论研究阶段。但由于这种进化方法能够很好地自动处理本体语义不一致问题，同时由于信念修正领域存在着大量成熟可行的算法、理论，因此这方面的研究有着比较好的发展前景。

3.4 本体变化传播技术

语义 Web 是一个巨型知识库，同时是一个“本体网”、“应用网”。本体作为语义 Web 的基础，在网络中是互相联系、互为重用的。另外，各种基于本体的应用也跟本体互相联系，不可分割。因此，在对本体进行修改后，本体变化的传播是一个极其重要的问题，否则“本体网”、“应用网”就会出现故障。

同时，本体变化的传播也是一个复杂的问题。在本体进行修改时，由于知识工程师无法完全预料到哪些本体和应用依赖于正在修改的本体，因此在修改完成后也就无法保证新的本体能够及时地被依赖的本体和应用发现，也无法保证相关本体与应用能够及时与新的本体相适应。在这种情况下，在客观上要求所有本体和应用能够接受本体变化传播变得很不现实。更为稳妥的办法是在主观上为其他本体和应用提供兼容的访问接口，从而能够自适应地处理外部需求。也就是说，对于已经接受了变化传播的外部请求，将访问请求自动转入新的本体；对尚未更新的外部请求，能够提供兼容的旧版本本体进行访问。

目前，本体版本机制[14]为主观的本体变化传播提供了一定的技术支持。所谓的本体版本机制，就是指在本体变化时，通过记录新版本的方式来记录变化，而不是覆盖旧版本。这样，在对待尚未及时更新的外部请求时，可以通过寻找兼容旧版本本体的方式来提供兼容的访问接口。另外，在发现有害或者不合理的变化时，可以快速地恢复原有的本体版本。

另外，本体在改变时，由于某些特殊的原因，本体的一致性可能无法得到保证。这时，变化依然需要传播到相关本体和应用中去。这时，不一致本体推理技术能够在一定程度上为外部请求提供有效的推理服务。

下面对本体版本机制和不一致本体推理技术进行简单的介绍。

3.4.1 多版本本体的表示方式

一般来说，本体多版本表示方式分为两种类型，即采用多个本体文件来表示多版本和采用单个本体文件来表示多版本。

对于第一种方式，本体在发生变化时，在对变化进行处理后不直接在原有本体上修改，而是创建一个新的本体，以供用户使用。版本之间变化的获取有多种方式，文[15]采用一个变化与标注本体(CHAO)来显式地记录本体变化的方式来让用户获知版本之间的改变，文[16]创建了一个 PROMPT 插件，可以使用 PROMPTDIFF 算法来比较两个本体版本的不同。

对于后者，则是采用虚拟版本[10]的方式。在本体修改时只保留一个基础本体版本，同时采用一种本体变化语言(CDL)来表示本体变化。通过基础版本和版本日志，可以推理得到后续的所有版本。文[10]中对本体变化语言和虚拟版本表示方式进行了详细的说明。

3.4.2 多版本本体推理技术

多版本本体推理通常需要解决的问题包括：语义变化日志查询、兼容性检查和本体版本选择等。在本体变化结果传播中，最为重要的是本体版本选择问题，即为外部的本体查询请求选择一个最为合适的版本来响应请求。

在文[17]中，作者提出了一个基于时态逻辑的多版本本体推理方法。在文中，作者在本体中扩充了多个时态算子，例如，$PreviousVersion\Phi$ 表示查询 Φ 在过去的版本中成立；$AllPriorVersion\Phi$ 表示查询 Φ 在之前所有的版本中均成立；$SomePriorVersion\Phi$ 表示查询 Φ 在之前的部分版本中成立。通过这些算法，可以组合成比较复杂的查询，从而找到合适的版本来应对请求。例如，在外部请求需要对某个查询 α 进行确认时，可以查询是否存在某个版本，使得 $PreviousVersion\alpha$ 成立，从而找到这个版本来提供服务。也可以查询 $SomePriorVersion\alpha$ 是否成立来判断是否存在一个版本，使得该查询成立。如果不存在，那么该查询在所有的版本中均不成立。时态算子的详细语义说明可以参见文[18]。

3.4.3 不一致本体推理技术

在某些特殊情况下，本体的一致性无法得到保证，或者本体中所存在的不一致暂时无法修复，但是仍然需要使用该本体来支持信息访问，这时就需要通过不一致本体推理技术来为外部请求提供服务。

在文[19]中，作者提出了一个线性扩展策略(Linear Extention Strategy)来应对针对不一致本体的查询。该策略的基本思想是，将不一致本体 O 的所有最大一致子本体(定义 3)抽取出来，并对这些子本体进行分析。如果存在一个一致子本体 O'，使得查询 α 在该本体中成立，而且对于所有的其他的一致子本体，$\neg\alpha$ 均不成立，那么就认为查询 α 在该不一致本体中是成立的。关于查询可靠性的证明可参见文[19]。

比较与总结 随着语义 Web 的发展，本体越来越多地深入到各项应用中。语义 Web 是一个复杂的动态环境，其中的知识是不断变化的。因此，作为其基础的本体也应该动态地调整，以适应环境的变化。本文以本体变化管理为中心，将本体变化管理划分为三个部分，即本体变化发现、本体变化处理和本体变化结果传播。同时，对目前与本体变化管理技术相关的研究内容进行了综述，对每项技术的功能、技术路线以及与本体变化管理的关系进行了分析。下面，我们将分别从这些技术所适用的阶段、功能、输入和输出四个方面，对这些技术进行统一的比较，如表 1 所示。

表1 本体变化管理技术比较

方法	适用的本体变化管理阶段	功能	输入	输出
本体调试	变化发现	在判断本体一致性的同时，找到不一致的根源	不一致本体	不一致的源头
本体模块化	变化发现、变化处理	对本体进行分块处理，使各分块逻辑上相对独立	本体	若干分块
本体进化	变化处理	对本体所要进行的改变以及改变所带来的不一致进行分析，维持本体语法和语义的一致性	本体、变化	维护了一致性的本体
本体版本机制	变化处理、变化传播	采用多版本的机制，为外部请求提供兼容的本体服务，并能够快速的将当前本体恢复为以往的版本	本体、变化、本体查询	多个版本的本体、一致的查询结果
不一致本体推理	变化传播	在本体一致性无法得到保证的情况下，为外部请求提供一致的服务	不一致本体、本体查询请求	一致的查询结果

总体而言，本体变化管理仍处于起步的阶段，目前针对它的研究散见于多个研究领域，例如本体进化、本体版本机制等，仍然缺乏一个统一的本体变化管理框架。统一的本体变化管理框架，能够真正以本体变化为核心，对本体变化进行全方位的管理，是一个值得注意的研究内容。

另外，在具体技术的研究上，也有许多需要注意的研究内容。例如，在本体进化方面，关于本体协同进化方法的研究仍然缺乏；在本体版本机制技术方面，如何利用现存的多个版本为相关应用提供有效的兼容服务的问题，也有许多可改进之处；在本体模块化方面，如何快速地对大规模本体的模块特性进行分析、本体的不一致在模块之间如何传播等问题，仍然需要进一步研究。

参 考 文 献

1 Berbes-Lee T, Handler J, Lassila O. The semantic Web. Scientific American, 2001, 284(5): 34～43

2 Fensel D. Ontologies: dynamics networks of meaning. In: Proceedings of the 1st Semantic Web working symposium. Stanford, CA, USA, 2001

3 Stojanovic L. Methods and Tools for Ontology Evolution. University of Karlsruhe, 2004

4 Haase P, Stojanovic L. Consistent Evolution of OWL Ontologies. In: Proceedings of the Second European Semantic Web Conference. Heraklion, Greece, 2005

5 Stojanovic L, et al. User-driven Ontology Evolution Management. In: European Conf Knowledge Eng and Management (EKAW 2002). Springer-Verlag, 2002

6 Parsia B, Sirin E, Kalyanpur A. Debugging OWL ontologies. In: 14th Intl Conference on World Wide Web. New York: ACM Press, 2005

7 Wang H, Horridge M, Rector A. Debugging OWLDL ontologies: A heuristic approach. In: 4th International Semantic Web Conference (ISWC2005). Galway, Ireland: Springer, 2005

8 Grau B C, Parsia B, Sirin E. Working with multiple ontologies on the semantic Web. In: Third Internatonal Semantic Web Conference (ISWC2004). Springer, 2004

9 Seidenberg J, Rector A. Web ontology segmentation: Analysis, classification and use. In: 15th International World Wide Web Conference. Edinburgh, Scotland, 2006

10 Plessers P, Troyer O D. Ontology Change Detection Using a Version Log. In: 4th International Semantic Web Conference, ISWC 2005. Galway, Ireland, 2005

11 Flouris G, Plexousakis D, Antoniou G. Evolving Ontology Evolution. In: SOFSEM06, 2006

12 Flouris G. On Belief Change and Ontology Evolution. In: Department of Computer Science, University of Crete, 2006

13 Alchourron C, Gärdenfors P, Makinson D. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. Journal of Symbolic Logic, 1985, 50: 510～530

14 Heflin J, Pan Z. A Model Theoretic Semantics for Ontology Versioning. In: Third International Semantic Web Conference(ISWC 2004). Springer, 2004

15 Noy N F, et al. A Framework for ontology evolution in collaborative environments. In: 5th International Semantic Web Conference(ISWC 2006). Springer, 2006

16 Noy N F, Musen M A. The PROMPT suite: Interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies, 2003, 59(6): 983～1024

17 Huang Z, Stuckenschmidt H. Reasoning with multiversion ontologies: a temporal logic approach. In: Fourth International Semantic Web Conference (ISWC2005). Springer, 2005

18 Huang Z, Stuckenschmidt H. Reasoning with Multi-version Ontologies. Department of Artificial Intelligence, Vrije Universiteit Amsterdam, 2005

19 Huang Z, Harmelen F V, Teije A T. Reasoning with inconsistent ontologies. In: the International Joint Conference on Artificial Intelligence(IJCAI'05). 2005