计算机科学 2003Vol. 30№. 7

# Web 仓储系统的视图刷新方案*)

张 岩[1] 杨冬青[2] 唐世渭[1]
(北京大学视觉与听觉信息处理国家重点实验室 北京 100871)[1]
(北京大学计算机科学技术系 北京 100871)[2]

## View-Refreshing Strategy in Web Repository

ZHANG Yan[1] YANG Dong-Qing[2] TANG Shi-Wei[1]
(National Laboratory on Machine Perception, Peking University, Beijing 100871)[1]
(Department of Computer Science and Technology, Peking University, Beijing 100871)[2]

**Abstract** Web repository utilizes materialized views to integrate information, so the most important task for it is to refresh the views periodically to keep its vitality. The goal of view refreshing is to obtain the maximal system freshness, however, most of traditional standards do not consider query effect so that the corresponding view-refreshing scheme is usually inefficient. In this paper, we first bring forward a more reasonable standard for system freshness, thereafter make a comprehensive discussion upon system refreshing frequency, system resource allocation and view-refreshing order. After analyzing the effect and function to system freshness for each factor, we give a general strategy for materialized view refreshing.

**Keywords** Web repository, Materialized view, View-refreshing, System freshness

## 1 Web 仓储的时新性标准

WWW 的迅猛发展使其成为全球信息传递与共享日益重要的信息资源。Web 仓储使用物化视图构建信息集成系统，是对 Web 信息进行充分利用的一种有效方法。Web 仓储具有高稳定性，查询速度非常快，非常适合决策分析等需要对信息进行深度加工的应用。与使用虚视图方法进行集成的系统不同，Web 仓储系统中的首要任务是物化视图的构建和维护，而视图刷新则是物化视图维护工作的主体。

Web 仓储视图刷新的目标是使系统获得最大的时新性(Freshness)。不同的系统时新性标准决定着不同的系统刷新方案。传统的系统时新性标准[2]认为系统中各视图的重要性是相同的，系统刷新方案主要考虑视图变化频率和系统刷新顺序等因素。但实际上，Web 仓储中各个视图具有不同的用户访问频率[4,6]，用户更关心查询返回的结果是否最新，而对于没有使用到的视图的情况并不关心。所以，访问频率高的视图显然具有更高的重要度，保持它们 up-to-date 会让用户更多地感到系统是"fresh"的。从用户使用效果的角度出发，我们可以给出如下的定义：

**定义**(Web 仓储系统的时新性) 物化视图 $v_i$ 在时刻 $t$ 的 freshness 为 $f(v_i;t)$，在$[t_0,t]$的访问频率为 $q_i$，那么 Web 仓储 WR 在时刻 $t$ 的 freshness 为：

$$F(WR;t_0,t)=\left(\sum_{i=1}^{N} f(v_i;t)*q_i\right)/\sum_{i=1}^{N} q_i$$

其中物化视图 $v_i$ 在时刻 $t$ 的 freshness 即 $f(v_i;t)$定义为：

$$f(v_i;t)=\begin{cases}1, & \text{if } v_i \text{ is up-to-date at time t}\\ 0, & \text{otherwise}\end{cases}$$

$F(WR;t_0,t)$的定义充分照顾到用户的查询效果，是一种面向用户的时新性标准。它额外的要求仅仅是视图查询频率，这是容易得到的。

**例 1** 一个典型的 Web 仓储系统中，假设系统中视图按查询频率可分为 3 个等级，系数分别为 1、3 和 10，每个等级中视图所占系统的份额分别为 70%、20%和 10%。如果我们不考虑查询频率，对每个视图都保持刷新频率和其变化频率相等，那么得到的系统时新性为 0.63。如果我们采取按查询频率同比分配刷新资源的方案，那么在获得同样时新性的情况下，系统可以减轻负担 18%。或者换一个角度，使用相同的资源，我们可以把系统时新性提高为 0.69。考虑到当系统刷新频率加倍时所获得的系统时新性才提高到 0.787，那么这种通过资源调配所获得的效果可以说是非常显著的。

在以下几节中，我们将依照 $F(WR;t_0,t)$定义的标准对系统刷新频率、系统资源分配以及视图刷新顺序等展开讨论，在明确了相关各因素对系统时新性的作用后，我们给出 Web 仓储中视图刷新的总体方案。

## 2 系统刷新频率

一般地说，如果系统的刷新能力比较强，使我们能经常刷新系统，那么它的时新性也就会比较高。然而，确定系统刷新频率是个复杂的问题，它涉及到系统刷新能力、刷新整个系统的代价、系统刷新时间的限制、网络带宽、涉及数据源的特点等等。这些因素必须综合考虑，才能决定系统的刷新能力，并非由一两个简单的算法就能决定。

系统刷新能力由系统处理能力、系统负载、网络带宽等决定，它通常是一个动态的数值，系统只能在不影响给用户提供正常使用的前提下进行刷新。至于刷新整个系统的代价，由于 Web 仓储中的物化视图在不断变化，同时每个视图的刷新代价也是动态的(由其内容和其它视图的关系以及数据源情况

*)国家重点基础研究发展规划(973)资助项目(G1999032705)。张 岩 博士，主要研究领域为数据库与信息系统。杨冬青 教授，博士生导师，研究领域为数据库与信息系统。唐世渭 教授，博士生导师，研究领域为数据库与信息系统。

综合决定),所以每次刷新时,系统刷新代价都会有所不同。一般,我们取最近几次的平均值作为系统刷新代价。粗略的描述可以如下给出:假设视图 $v_i$ 的刷新代价为 $r_i$,系统一天刷新了 $k$ 个视图,那么系统的刷新能力为:

$$R_s=\sum_{i=1}^{k} r_i$$

我们可以取系统在一段较长时间内的刷新能力的平均值作为基准。而系统的刷新频率则可以粗略地按如下定义给出:

$$f_{system}=R_s/\sum_{v_i\in WR} r_i$$

实际上,系统刷新频率更多地是一个经验值,我们可以通过较长时期的观察得到一个较稳定的取值区间,同时可以通过系统处理能力的调配使其稳定在一个我们需要的范围内。

## 3 系统刷新中的资源分配

### 3.1 问题和数学模型

确定系统的刷新能力后,我们需要进一步考虑系统刷新资源如何分配给各个视图。也就是说,我们要考虑当系统刷新能力一定时,如何分配视图的刷新频率,才能使系统获得最大的时新性。这个问题可进一步明确为:对于 Web 仓储 $WR$ 中的视图 $v_i$,给定其查询频率 $q_i$、刷新复杂度 $r_i$、变化频率 $\lambda_i$,同时我们知道系统的刷新能力 $R_s$,那么如何确定刷新频率 $f_i$,可以获得最大的系统时新性?

首先,我们要计算视图 $v_i$ 在给定变化频率 $\lambda_i$ 和刷新频率 $f_i$ 的情况下可以获得多大的时新性;然后对 $q_i$ 加权,得到 $WR$ 中所有视图的时新性总和。系统资源分配的任务就是使这个总和最大化,同时保持刷新总复杂度 $\sum r_i \cdot f_i$ 在系统刷新能力 $R_s$ 的控制下。

我们需要合适的数学模型。我们知道,Web 仓储中的物化视图来源于一处或多处 Web 页面数据,而 Web 页面的变化遵循 Poisson 过程[1,3];根据文[8],多个 Poisson 过程的合成仍是一个 Poisson 过程,其变化频率为各变化频率的和。这样,我们就使用 Poisson 过程作为数学模型,来描述各视图的变化。

按 Poisson 模型,视图 $v_i$ 在时间 $(s,s+t]$ 不发生变化的可能性为 $e^{-\lambda_i t}$,其发生变化的可能性为 $1-e^{-\lambda_i t}$,那么根据视图时新性的定义,该视图的期望时新性(Expected freshness)为:

$$E[(F(v_i;)]=0\cdot(1-e^{-\lambda_i t})+1\cdot e^{-\lambda_i t}=e^{-\lambda_i t}$$

设视图 $v_i$ 的刷新周期为 $I_i$,那么有 $I_i=1/f_i$,其中 $f_i$ 为其刷新频率。我们记 $\overline{F}(v_i)$ 为该视图的平均时新性,那么有:

$$\overline{F}(v_i)=\frac{1}{I_i}\int_0^{I_i} e^{-\lambda_i t}dt=\frac{1}{I_i}*\frac{1-e^{-\lambda_i I_i}}{\lambda_i}=\frac{1-e^{-\lambda_i/f_i}}{\lambda_i/f_i}$$

记 $\overline{F}(WR)$ 为 Web 仓储 $WR$ 的时新性总和,则有:

$$\overline{F}(WR)=\sum_{v_i\in WR}\overline{F}(v_i)*q_i,$$

限制条件为:

$$\sum_{v_i\in WR} r_i\cdot f_i\leqslant R_s$$

这样,系统刷新资源分配问题就可以正式地表述为:找到一组 $f_i$,使其在满足 $\sum r_i\cdot f_i\leqslant R_s$ 的条件下,使 $\overline{F}(WR)$ 最大化。

这个问题是非常复杂的。首先,我们无法用纯数学的方法给出其精确解;其次,即使我们对 $f_i$ 加一些限制,比如限定它是带一位或两位的小数,那么该问题仍然是 NP 完全问题。这样,我们就需要从实用的角度出发,寻求在合理时间内(如幂次较低的多项式时间)能够给出合理解(经实验可用的有意义解)的近似解法。

### 3.2 视图刷新频率对时新性的影响

视图刷新频率越高,视图所获得的时新性就会越高。但它们是否就是一种线性关系呢?通过对计算公式 $\overline{F}(v_i)=(1-e^{-\lambda_i/f_i})/(\lambda_i/f_i)$ 的观察可以发现,刷新频率 $f_i$ 加快时,视图时新性 $\overline{F}(v_i)$ 的值并非成比例增大,而是非常缓慢地增大(见图1)。对于变化频率太快的视图,即使我们多刷新了几次,它的时新性也没有太大变化,所以我们不应该花特别多的资源在它上面,而应该转向其它相对较稳定的视图。

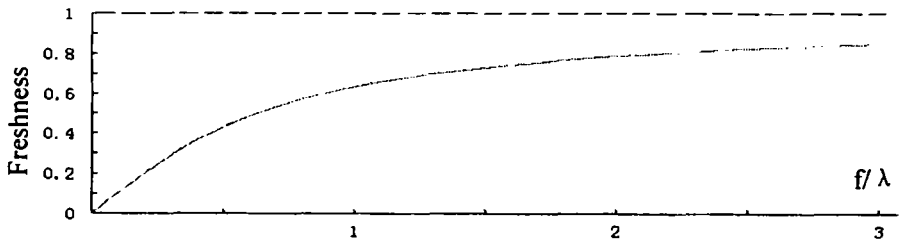


图 1 刷新频率与数据时新性

**例 2** 记 $t=f/\lambda$,按固定顺序刷新得到的系统时新性 $SF=(1-e^{-1/t})*t$。我们考察 $SF$ 的微分可以发现,$SF'(1)=0.264$,$SF'(2)=0.090$,$SF'(3)=0.045$ 以及 $SF'(4)=0.026$。由此说明,当系统刷新频率小于数据变化频率 $\lambda$ 时,刷新频率的提高对系统时新性的影响非常明显;当系统刷新频率在 $(\lambda,2\lambda)$ 之间时,刷新频率的提高对系统时新性的影响趋于平和;当系统刷新频率大于 $2\lambda$ 时,要想获得系统时新性的提高,刷新频率必须大幅提高才行。

### 3.3 视图变化频率的影响

视图变化频率对视图刷新频率有着很大的影响:一般地说,视图变化得越快,其刷新频率也应该越高,这样才能保持时新性。然而,正像我们前面看到的那样,当刷新频率达到一定程度后,所获得的数据时新性只是在非常缓慢地增大,这时我们还不如把资源多用在那些变化比较稳定的视图上。

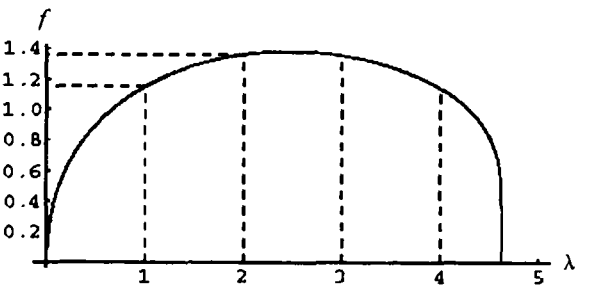


图 2 刷新频率对变化频率的最大时新性解

**例 3** 假设 $WR$ 中有 5 个视图，它们的变化频率分别为 1,2,3,4,5 次/天。$WR$ 的刷新能力为 5 视图 * 次/天，那么通过 Lagrange 乘子的方法可以得到每个视图每天刷新的最佳次数，按顺序分别为：1.15,1.36,1.35,1.14,0.00 次。此时系统时新性为 0.374。如图 2 所示，其中 $f$ 为视图刷新频率。

### 3.4 视图查询频率的影响

如果视图查询频率不同，那么我们就必须考虑查询频率带来的影响：查询频率越高，视图对系统时新性的影响就越大。下面的例子分析了查询频率在视图刷新中的相对作用。

**例 4** 假设 $WR$ 中有 2 个视图 $V_1$ 和 $V_2$，它们的刷新代价和变化频率都相同，但后者的查询频率为前者的 $n$ 倍，即 $n=q_2/q_1$。那么当系统刷新能力为 2 视图/天时，刷新频率 $f_1$ 和 $f_2$ 的最优解可以使用 Lagrange 乘子方法获得，如表 1。

表 1 查询频率对最优刷新频率的影响

| | $N=1$ | 2 | 3 | 4 | 5 | 6 | 7 | 11.09 |
|---|---|---|---|---|---|---|---|---|
| $f_1$ | 1 | 0.75 | 0.61 | 0.51 | 0.43 | 0.37 | 0.31 | 0 |
| $f_2$ | 1 | 1.25 | 1.39 | 1.49 | 1.57 | 1.63 | 1.69 | 2 |
| $f_2/f_1$ | 1 | 1.67 | 2.28 | 2.92 | 3.63 | 4.41 | 5.45 | — |
| $n^{0.8}$ | 1 | 1.74 | 2.41 | 3.03 | 3.62 | 4.19 | 4.74 | 6.85 |

从表 1 中可以看出，当 $n\leqslant 6$ 时，$f_2/f_1$ 值和 $n^{0.8}$ 接近。而当 $n\geqslant 11.09$ 时，刷新频率为 0，即所有的资源都用于 $V_2$ 的刷新。

### 3.5 视图刷新复杂度的影响

一般地说，刷新复杂度高的视图会占用更多的系统资源，所以系统在同样条件下更愿意刷新那些复杂度低的视图。

**例 5** 我们考察刷新代价(刷新复杂度)$r_i$ 对视图刷新的影响。假设 $WR$ 中有 2 个视图 $V_1$ 和 $V_2$，它们的查询频率和变化频率都相同，但前者的刷新代价 $r_1$ 为 1，后者的刷新代价 $r_2$ 为 $n$。那么当系统每天刷新能力为 2 时，刷新频率 $f_1$ 和 $f_2$ 的最优解如表 2 所示。

表 2 视图刷新代价对最优刷新频率的影响

| | $N=1$ | 2 | 3 | 4 | 5 | 6 | 7 | 11 |
|---|---|---|---|---|---|---|---|---|
| $f_1$ | 1 | 0.95 | 0.995 | 1.09 | 1.224 | 1.368 | 1.51 | 2 |
| $f_2$ | 1 | 0.53 | 0.335 | 0.228 | 0.155 | 0.105 | 0.07 | 0 |
| $f_2/f_1$ | 1 | 1.79 | 2.97 | 4.78 | 7.90 | 13.0 | 21.6 | — |
| $3^{n/e}$ | 1.496 | 2.24 | 3.35 | 5.04 | 7.54 | 11.3 | 16.93 | 85.26 |

通过分析看出，在很大的范围内，使用 $3^{n/e}$ 逼近 $f_2/f_1$ 很大程度上都比较可行。

### 3.6 系统资源分配问题的近似解法

通过大量的数据分析和函数逼近，综合考虑视图查询频率、变化频率和刷新代价对视图刷新频率的影响，它们给出下面的近似解法。它需要的时间仅为 $O(n)$，效率非常高；同时通过实验验证，它获得原系统时新性也较好。

Algoeirhm SRS(Simplified-Refresh-Set)：

Step1：计算 $\bar{\lambda}=\dfrac{\sum \frac{q_i^{0.8}}{3^{r_i/e}}\cdot\lambda_i}{\sum \frac{q_i^{0.8}}{3^{r_i/e}}}$；

Step2：计算 $w_i=\sum \frac{q_i^{0.8}}{3^{r_i/e}}(\bar{\lambda}-|\bar{\lambda}-\lambda_i|)^{1/2}$；

Step3：计算 $W_r=\sum(w_i\cdot r_i)$

Step4：计算 $f_i=\dfrac{K_e\cdot w_i}{W_r}$

Web 仓储系统的刷新是一个复杂的过程，在无法以多项式时间给出其最优解法的情况下，给出一些计算代价较低的近似解法是很有意义的。

## 4 系统中的视图刷新顺序

除了刷新频率，视图的刷新顺序在系统刷新方案中也是必须考虑的要素。一般来说，视图的刷新顺序有 3 种：固定顺序刷新(fixed-order，每次循环都按相同的顺序刷新所有视图)、循环随机顺序刷新(random-order，每次循环都刷新所有的视图，但每次的刷新顺序随机决定)和随意选取刷新(purely random，每次随意选取一个视图刷新，不知何时才能刷新所有视图)。不同的系统资源分配方案中，这 3 种刷新顺序所起的效果也不尽相同。为了简单起见，我们假设所有的视图具有平等的地位(都以相同的频率进行变化，视图的更新复杂度相同，它们的查询频率也都相同)，这时系统对所有视图都采取相同的刷新频率 $f$。这种情况下，按固定顺序刷新效果最好(文[2]中已证明)，而循环随机刷新效果次之，纯粹的随意选取刷新效果最差。其时新性趋势对比见图 3，其中 $r=\lambda/f$。

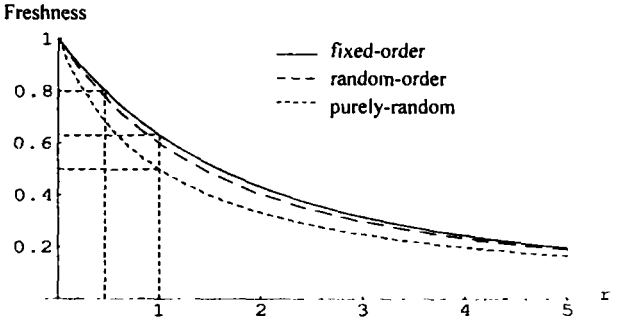


图 3 不同刷新顺序的系统时新性趋势图

当系统中各视图的查询频率、刷新复杂度和变化频率不同时，各种刷新顺序所产生的效果更加复杂。通过明确的公式结果对一般情况进行分析非常困难，但通过一些诸如上述特殊情形的分析，我们有理由相信固定顺序刷新可以获得很好的效果。

## 5 视图刷新的总体方案

从前面的讨论可以看出，视图刷新是一个复杂的问题。一方面，系统需要通过刷新保持其生命活力；另一方面，由于资源的限制，系统不能过于频繁地刷新。所以，需要统筹考虑视图查询频率、视图变化频率、视图刷新代价和系统刷新能力，并且刷新方案要切实可行，具有较好的可操作性。综合以上讨论，我们给出视图刷新的总体方案如下：

(1)把系统中的物化视图按其访问规律分为若干个不同的分区，使每个分区都在自己的使用波谷时间进行刷新。视图属于哪个分区，可以定期或随时进行动态调整。

(2)把系统中的物化视图按算法 SRS 求出其刷新频率。

(3)按刷新频率组织相应的刷新序列：序列中视图的刷新次数与其刷新频率成正比；顺序按视图查询频率由高到低排列，查询频率相同的视图则按变化频率由低到高排列；视图在刷新序列中多次出现时，位置尽量平均分布。

(4)刷新序列中的视图按顺序刷新：如果系统较忙，待刷新的视图又具有较高的刷新复杂度，那么把该视图放入延迟队列(先进先出)，刷新下一个视图。

(5)延迟队列的处理：每次刷新时先检查延迟队列，如果其队首视图可以被刷新，那么就刷新该视图，否则把该视图移入队尾，然后检查下一个视图。延迟队列都被检查一遍后，开

始刷新普通队列中的视图。

以上只是一个初步的 Web 仓储系统视图刷新原则。实际上，还有许多因素需要进一步考虑，如维护数据一致性时需要对系统进行关联性分区，不同分区的视图并行刷新，每个分区内部再如上考虑。另外，由于返回给用户的查询结果不一定都是最新内容，所以用户希望数据的"过期时间"越短越好。这里就有一个数据年龄的概念。一般地说，物化视图的年龄等于其基础数据发生变化后至今的这段时间，而系统的年龄等于视图年龄按查询频率的加权平均值。若考虑数据年龄，则 Web 仓储系统视图刷新的目标就变为"在系统刷新能力和刷新时间的限制下，希望以较小的代价，获得最大的系统时新性，并且使系统年龄尽量小"。

**结论** 本文介绍了 Web 仓储在系统维护的过程中如何进行物化视图的刷新，并给出了相关算法。这些算法已经在我们的 973 项目的原型系统中得到初步使用和验证，取得了较好的效果。未来的工作，除了需要在我们的系统中进一步发展和完善相关算法外，还希望能够在大型、实用的 Web 集成环境中验证算法的可靠性和有效性。

## 参 考 文 献


1 Brian E. Brewington and George Cybenko: How dynamic is the Web? Computer Networks, 2000, 33: 257～276

2 Cho J, Garcia-Molina H. Synchronizing a database to improve freshness. In: Proc. 2000 ACM Int. Conf. on Management of Data (SIGMOD), May 2000

3 Cho J, Garcia-Molina H. Estimating frequency of change. Submitted for publication, 2000

4 Cho J. Crawling the Web: Discovery and Maintenance of a Large-Scale Web Data: [Ph. D. thesis]. Stanford University

5 Hirai J, Raghavan S, Garcia-Molina H, Paepcke A. WebBase: A Repository of Web Pages. In: Proc. of 9th Int. World-Wide Web Conf., 2000, 277～293

6 Liu Haifeng, Ng W-K, Lim E-P. Keeping a very large website up-to-date: Some feasibility results. In: Proc. of the 1st Intl. Conf. on Electronic Commerce and Web Technologies (EC-Web2000), Greenwich, UK, Sep. 2000

7 George B, Thomas Jr. Calculus and analytic geometry. Addison-Wesley, 4th edition, 1969

8 Taylor H M, Karlin S. An Introduction To Stochastic Modeling. Academic Press, 3rd edition, 1998

9 Xyleme L. A dynamic warehouse for XML data of the Web. IEEE Data Engineering Bulletin, 2001


（上接第 17 页）

决定构件服务是否可用；在构件服务执行的时候，当发生个别构件服务"失效"的情况下，构件服务提供者要能够向居中 Agent 进行服务查询，并和替代构件服务的服务提供者进行协商。

框架中的第三类协议是构件服务的调用协议。构件之间彼此能够通信是跨越 Internet 使用构件服务的基本要求。文[8]对主流的构件规范中存在"笨拙"总线的问题进行了详细的分析，提出采用移动 Agent 来替换传统的基于 RPC 的"笨拙"总线。但是，由于移动 Agent 对客户端和服务器端的软件环境都有特殊的要求，而且涉及到很多安全问题及多种通信协议，所以实现将需要复杂的环境支持[9,10]。文[9]中实现了基于 SOAP(Simple Object Access Protocol)的服务代理结构，可以完成基于移动 Agent 的对象间异步通信方式，而且 SOAP 已经是业界接受的标准，所以在构件服务框架中我们采用 SOAP 作为构件服务间相互调用的标准协议。

**结论** 本文提出了一个 Internet 上的构件服务框架，它具有如下优点：①框架中的各软件体基于 Internet 上通用的协议及标准进行协同工作，使得框架的设计及实现简单；②该框架模型可以充分与人们对软件的需求相适应，为人们便利地获取各种构件提供的服务以及改善软件重用提供了新的方法。

## 参 考 文 献


1 Spinellis D. Explore, Excogitate, Exploit: Component Mining. IEEE Computer, 1999, 32(9): 114～116

2 Mili A, Mili F, Mili A. Reusing Software: Issues and Research Directions. IEEE Transactions on Software Engineering, 1995, 21(6): 528～562

3 Arora S. Probabilistic Checking of Proofs and Hardness of Approximation Problems, 1995: [A technical report based on the author's Ph. D thesis]. Contains a proof of the PCP theorem

4 Martin J. Web Services: The Next Big Thing, xml Journal 2. http://www.sys.com/ xml

5 Myerson J M. Web Service Architecture. http://www.webservicesarchitect.com/content/articles/webservicesarchitectures.pdf

6 Sollazzo T, et al. Semantic Web Service Architechure-Evolving Web Service Standards toward the Semantic Web. http://www.aifb.uni-karlsruhe.de/WBS/sha/papers/swobis_flairs.pdf

7 McIlraith S, Son T C, Zeng H. Semantic Web Services. IEEE Intelligent Systems, Special Issue on the Semantic Web, 2001, 16(2): 46～53

8 LU Jian, ZHANG Ming, LIAO Yu, TAO Xian-ping. Research on Componentware Framework Based on Mobile Agent Technology. Journal of Software, 2000, 11(8): 1018～1023

9 Shin Nakajima. A SOAP-based Infrastructure for Service Broker. http://pizza.cs.ucl.ac.uk/ xse01/ready/5.pdf

10 Wooldridge M J, Jennings N R. Pitfalls of Agent-Oriented Development. In: Proc 2nd Int. Conf. on Autonomous Agents (Agents-98), Minneapolis, USA, 1998. 385～391