一种改进的基于路径的构件软件可靠性模型研究

张 伟 张为群
(西南大学计算机与信息科学学院 重庆 400715) (重庆市智能软件与软件工程重点实验室 重庆 400715)

摘 要 基于构件的软件可靠性分析往往把构件的可靠性当作自身固有不变的属性,忽略了在不同的运行路径下因构件所处的交互环境不同造成的实际可靠性的变化。因此,提出一种改进的基于路径的构件软件可靠性模型,它引入构件动态迁移图来建立构件软件运行路径和构件可靠性关系,给出一种改进的基于路径的用以计算构件软件可靠性的方法。最后通过实例验证了该方法的有效性和可行性。

关键词 构件软件,软件可靠性,动态迁移图,TE-NHPP

Study of Improved Component-based Software Reliability Model Based on Route

ZHANG Wei ZHANG Wei-qun
(Faculty of Computer and Information Science, Southwest University, Chongqing 400715, China)
(Chongqing Intelligent Software and Software Engineering Laboratory, Chongqing 400715, China)

Abstract Component-based software reliability analysis typically takes the reliability of component as an invariable property of component itself, while disregards the fact that it is changed when the component context that it interacts with is changed due to a different route. This paper proposed a improved component-based software reliability model based on route, which introduces the dynamatic transition graph to build the relationship between the route of component-based software and component reliabity, and gave a improved method based on route to caculate the componet-based software reliability. At last, the result of case demonstrates the feasibility and validity of the approach.

Keywords Component-based software, Software reliability, Dynamatic transition graph, TE-NHPP

1 引言

随着面向对象的系统设计和基于 Web 的软件开发的广泛应用,构件软件正迅猛发展着。开发者可以通过组装已有的构件来开发新的应用系统,从而达到软件复用的目的,而软件构件技术是软件复用的关键因素。构件可以是第三方构件、自主研发的构件或合约研发的构件,传统构件软件可靠性模型将构件可靠性当作构件软件中可替换的、独立于软件的且相互独立的"插件"对待,把构件可靠性当作一种仅依赖于构件本身的静态属性。实际上,由于构件开发者和构件使用者相互分离,使整个系统开发在一个异构环境下[1],构件的使用环境可能会出现较大差异,构件的可靠性也会发生变化,在不同的系统操作剖面下,构件之间的转移概率发生变化,这些因素直接影响到系统可靠性合成过程中结果的准确性。

本文利用改进的 TE-ZHPP 模型来度量构件的失效密度函数,提出了一种改进的基于路径的软件可靠性模型。此模型利用构件迁移情况,构建构件的迁移图,建立构件软件的八元序列描述模型,结合构件的失效密度函数,计算构件的可靠性,其改变了传统的将构件可靠性作为定值的计算方法,同时结合基于路径的方法,计算每条运行路径的可靠性,利用路径的可靠性计算构件软件的可靠性。该模型既考虑系统操作剖面对构件可靠性影响,又考虑构件软件的每条运行路径对构件软件的可靠性影响,同时也考虑构件软件的体系结构,符合基于构件软件及软件开发的特征,具有较广的覆盖面和较强的兼容性。

2 构件的特征

构件是指封装了数据和功能,在运行时能够通过参数进行配置的模块[2]。

定义 1 构件表示为一个四元组 $C=(No,PORT,D,R)$,其中:

(1) No 表示构件名称,作为构件的唯一标识。

(2) $PORT=\{port_1,port_2,\cdots,port_k \mid k\in N \mid \wedge k\geqslant 1\}$ 是构件的端口集合。

假设 $PORT$ 为构件 C 的端口集合[3],则满足条件 $\sum_{port_i\in PORT} f(port_i)=1$ 的函数 $f: port_i \rightarrow p$ 就是构件 C 的一个使用模型,其中 $p\in[0,1]$ 为出现的概率。

(3) $D=\{d_1,d_2,\cdots,d_k \mid k\in N\wedge k\geqslant 1\}$ 是构件端口参数配置集合,$d_i(1\leqslant i\leqslant k)$ 是端口 $port_i(1\leqslant i\leqslant k)$ 具体参数配置描述。

其中参数配置描述由 DateType-Def, Port-Beha-vior 构

到稿日期:2010-03-02 返修日期:2010-06-13 本文受重庆市自然科学基金项目(CSTC,2006BA2003)资助。

张 伟 男,硕士生,主要研究方向为软件工程、构件软件、软件可靠性,E-mail:scandrey@126.com;张为群 男,教授,主要研究方向为软件工程、形式语言、软件测试。

成。DateType-Def 是端口消息交互的数据类型定义；Port-Behavior 是端口功能、行为的描述。

(4)$R\in[0,1]$表示构件可靠性，存在函数 $g:t\to R$，其中 t 表示时间。

由于构件开发者与构件使用者的分离，基于构件软件的可靠性评测带来新的挑战：构件开发者不能预知构件具体的使用环境；构件使用者无法明了构件的内部细节。事实上，来自理论分析和实践经验的证据表明：在构件软件中，相比其它因素，构件可靠性对系统可靠性的影响更大[4]。因此，考虑如何刻画函数 g 来提高模型评估的准确性是研究的重点和难点，本文函数 g 就是构件失效密度函数。

3 TE-NHPP 估算构件失效密度函数

构件失效密度函数是在时刻 t 构件单位时间内的失效数。估算构件失效密度可以利用一些传统的软件可靠性模式，如 JM 模型、超几何模型、NHPP 类模型[5-7]。估算构件失效密度采用的方法要依对构件的了解程度、构件的类型和测试数据而定[8]。Kanoun 和 Sabourin[9] 在非齐次 Passion 过程基础上提出一种模型用于估计构件的平均失效率，Gakhale[10] 利用加强的 NHPP 模型确定构件的失效率，Everett[11] 利用构件的各种静态和动态特性以及每一个构件的使用建立了一个所谓的 EET 模型用于估计构件的失效密度。而本文 TE-NHPP 模型估计构件失效密度函数是基于 NHPP 模型的。

模型中函数定义：$M(t)$ 表示为 $[0,t)$ 时间段内检测出的累积故障数；M 表示构件系统中潜伏的故障总数的初始值；$r(t)$ 为 NHPP 模型估算出的构件失效密度函数；$\lambda(t)$ 为 TE-NHPP 估算出的构件失效密度函数。

3.1 模型假设

(1)对任意选定的正整数 n 和任意选定的 $0\leqslant t_0\leqslant t_1\leqslant t_2\leqslant\cdots\leqslant t_n$，则 n 个增量 $M(t_1)-M(t_0),M(t_2)-M(t_1),\cdots,M(t_n)-M(t_{n-1})$ 相互独立；

(2)对软件测试过程中的故障计数过程建模。设软件测试时，故障的出现用计数过程 $\{M(t),t\geqslant 0\}$ 表示，$\{M(t),t\geqslant 0\}$ 符合非齐次柏松分布，失效密度函数为 $r(t)$。同时，$\{M(t_0,t)=k\}$ 事件发生概率为：

$$P\{M(t_0,t)=k\}=\frac{[r(t-t_0)]^k}{k!}e^{-r(t-t_0)} \tag{1}$$

(3)累积错误数的期望函数 $M(t)$ 满足边界条件：

$$M(t)=0,\lim_{t\to\infty}M(t)=M$$

由模型假设可以得知 TE-NHPP 是基于 NHPP 的非齐次柏松分布，因此：

$$M(t)=M(1-e^{-bt}) \tag{2}$$

式中，b 为故障检率的初始值，故障检测率表示软件内残存一个差错的差错发现率，故障检测率的大小可以说明任意时刻下发现软件内残存差错的难易度。而时刻 t 构件单位时间内的失效数即构件失效密度函数为：

$$r(t)=\frac{dM(t)}{dt} \tag{3}$$

而对于这些函数求解可以利用测试数据结合最小二乘法，本文不做叙述。

3.2 TE-NHPP

上述模型假设是基于 NHPP 模型的，而 NHPP 模型仅仅使用测试用例基于时间来对被测构件的失效率进行度量，忽略了其它一些十分重要的影响因素，如测试用例有效性、测试环境等。TE-NHPP 模型将测试有效性加入 NHPP 模型得到构件失效密度函数，从而提高构件失效率的精度。为了得到测试有效性，可以采用故障注入法，通过计算一个测试集所检测到的故障数与注入故障总数比，就可以估算测试用例的有效性。

定义 2 测试用例有效性 a[12] 定义为被测构件在给定测试时间 t 内，由测试用例发现的故障数与注入故障总数之比，即：

$$a=f(t)/N \tag{4}$$

式中，$f(t)$ 是给定测试时间 t 内所发现的注入故障数，N 是注入故障总数，从式(4)可以得出，测试用例有效性应该在 $[0,1]$ 之间，a 的值越大测试用例就越有效。在计算测试集有效性时利用到的测试用例集与测试期间检测累积错误数利用到的测试用例集是一致的。

Algorithm 1 TE-NHPP(NON-HomogenEous Possion Process Based on Test Efficiency)

Input：测试用例集 M，N 个故障，$r(t)$

Output：$\lambda(t)$

Step1 将 N 个故障注入至被测程序中，$b=N,i=0,f(t)=0$；

Step2 使用测试用例 m_i 测试被测程序，当检测出一个故障后 $f(t)++$，然后将程序恢复至原来的状态，$M=M-\{m_i\},b--,i++$；

Step3 若 $b==0$，则 $a=1$，转至 Step5；

Step4 若 $M!=\Phi$ 转至 Step2；否则计算 $a=f(t)/N$；

Step5 计算构件失效密度函数 $\lambda(t)=r(t)/\sqrt[4]{a}$。

4 改进基于路径的构件软件可靠性模型

对于构件软件的可靠性，既要考虑构件自身的可靠性又要考虑构件软件的体系结构。目前构件软件系统的可靠性评估方式一般有 3 种形式[13]：基于路径的模型、基于状态的模型和基于运行剖面的模型。基于路径法通常在系统构建完毕后，针对测试用例所遵循的路径，计算该路径上各构件的使用数量和概率来计算软件的可靠性；状态法把构件软件系统的执行过程当作构件之间的状态转移过程，利用随机过程理论，构造 Markov 链，对构件系统可靠性进行计算；运行剖面是对构件软件系统输入域的刻画，针对不同的输入域、不同的上下文，单一构件的可靠性也不一致，计算不同的剖面出现概率、构件之间的迁移概率等来得到系统的可靠性。

4.1 构件动态迁移图

定义 3 构件 c_i 迁移到构件 c_j，则表示为 (c_i,c_j)。

定义 4 构件动态迁移图 G 是一个八元序列 $\langle C,T,S,E,P,R,L,Time(c_i,l_j)\rangle$，其中 C 为组成系统的构件的集合；T 为构件间的迁移集合，由定义 3 刻画；$S\subseteq C$ 为起始构件集合，$E\subseteq C$ 为终止构件集合，P 为构件间的迁移概率集合，存在函数 $h:t\to p$；R 为构件失效密度函数集合，L 为运行路径集合，$Time(c_i,l_j)$ 为构件 c_i 在运行路径 l_j 中平均执行时间。

定义 5 在构件动态迁移图 $G=\langle C,T,S,E,P,R,L,Time(c_i,l_j)\rangle$ 中，(c_i,c_j) 的迁移概率 p_{ij} 为：

$$p_{ij}=\begin{cases}1, & (c_i,c_j)\notin T\\ \dfrac{n(c_i,c_j)}{\sum_{j=0}^{|C|}n(c_i,c_j)}, & (c_i,c_j)\in T\end{cases} \tag{5}$$

式中，$n(c_i,c_j)$表示(c_i,c_j)迁移次数。

定义 6 在构件动态迁移图 $G=\langle C,T,S,E,P,R,L,Time(c_i,l_j)\rangle$中，若存在 $c_0t_0c_1t_1\cdots c_nt_n$，满足：

(1)$0\leqslant i\leqslant n, c_i\in C$，

(2)$0\leqslant i\leqslant n, t_i\in T$，

(3)$c_0\in S, c_n\in E$，

则称该路径为构件动态迁移图 G 的一个运行路径。

定义 7 若构件 c_i 在运行路径 l_j 中，则表示为 $c_i\in l_j$；若构件 c_i 不在运行路径 l_j 中，则表示为 $c_i\notin l_j$。

定义 8 在构件动态迁移图 $G=\langle C,T,S,E,P,R,L,Time(c_i,l_j)\rangle$中，若 $c_i\in l_j$，则构件 c_i 在路径 l_j 中的可靠性为：

$$R(c_i,l_j)=\begin{cases}e^{-\int_0^{Time(c_i,l_j)}\lambda(t)dt}, & c_i\in l_j\\ 1, & c_i\notin l_j\end{cases}\tag{6}$$

式中，$\lambda(t)$为构件 c_i 的失效密度函数，由式(6)可以发现构件的可靠性是一个变化的值，在不同运行路径中，构件的可靠性是不一样的。而构件动态迁移图是建立构件失效密度函数和构件可靠性关系的桥梁。

4.2 构件软件可靠性

在讨论构件可靠性之前，先做如下假设：较大规模的连接件可以抽象为独立的构件参加到构件软件中，剩下较小规模的连接件则假设其足够简单和可靠。

定义 9 在构件动态迁移图 $G=\langle C,T,S,E,P,R,L,Time(c_i,l_j)\rangle$中，若存在 $c_0t_0c_1t_1\cdots c_nt_n$ 运行路径 l_j，则此路径的出现概率为：

$$F_{l_j}=\prod_{t=1}^{|C_i|}\prod_{s=1}^{|C|}p(c_s,t_t)\text{且}(c_s\in l_j\wedge c_t\in l_j)\tag{7}$$

式中，$p(c_s,c_t)$为构件 c_s 到构件 c_t 的迁移概率。

定义 10 在构件动态迁移图 $G=\langle C,T,S,E,P,R,L,Time(c_i,l_j)\rangle$中，若存在 $c_0t_0c_1t_1\cdots c_nt_n$ 运行路径 l_j，则此路径的可靠性为：

$$R_{l_j}=\prod_{i=1}^{|C|}R(c_i,l_j)\text{且 }c_i\in l_j\tag{8}$$

式中，$R(c_i,l_j)$为构件 c_i 在运行路径 l_j 中的可靠性。

因此，一个构件软件其构件动态迁移图 $G=\langle C,T,S,E,P,R,L,Time(c_i,l_j)\rangle$，此系统的可靠性为：

$$R=\sum_{l_i\in L}(R_{l_i}*F_{l_i})/\sum_{l_i\in L}F_{l_i}\tag{9}$$

5 实例分析

以一简单的构件应届生求职系统为例，计算系统可靠性，包括 c_1(登陆注册)、c_2(查询已审记录)、c_3(修改已审记录)、c_4(提交新的申请)、c_5(退出)这 5 个简单构件(这里将功能模块理解为构件)。其构件动态迁移图粗略表示为图 1。

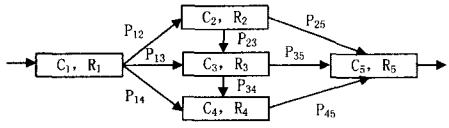

图 1 求职系统构件动态迁移图示例

图中，c_1 表示起始状态，c_5 表示终止状态，共存在 6 条运行路径为 $l_1=(c_1,c_2,c_5)$，$l_2=(c_1,c_2,c_3,c_5)$，$l_3=(c_1,c_2,c_3,c_4,c_5)$，$l_4=(c_1,c_3,c_5)$，$l_5=(c_1,c_3,c_4,c_5)$，$l_6=(c_1,c_4,c_5)$。每个构件在每条路径的可靠性集合为：

$$R=\begin{bmatrix}0.9231 & 0.9456 & 0.8561 & 0.8932 & 0.9635 & 0.8765\\ 0.8256 & 0.8456 & 0.7695 & 1 & 1 & 1\\ 1 & 0.9128 & 0.8963 & 0.8791 & 0.9432 & 1\\ 1 & 1 & 1 & 1 & 0.7963 & 0.8697\\ 0.8431 & 0.9418 & 0.7638 & 0.8476 & 0.9514 & 0.7693\end{bmatrix}$$

其迁移概率集合分别为：

$$P_1=\begin{bmatrix}1 & 0.1 & 0.4 & 0.5 & 1\\ 1 & 1 & 0.4 & 1 & 0.6\\ 1 & 1 & 1 & 0.5 & 0.5\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

$$P_2=\begin{bmatrix}1 & 0.3 & 0.3 & 0.4 & 1\\ 1 & 1 & 0.5 & 0 & 0.5\\ 1 & 1 & 0 & 0.2 & 0.8\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

用本文改进的基于路径的构件软件可靠性模型，可以得到模型的情况对比，如表 1 所列。

表 1 模型的情况对比

构件在每条路径的可靠性集合	迁移概率集合	系统可靠性
R	P_1	0.6253
R	P_2	0.6280

通过此实例，可以看出：1. 此模型中构件可靠性与构件使用环境、上下文等因素有关，构件可靠性不是一个固定值，在不同运行路径中，因为执行时间不同会导致其可靠性不同。2. 此模型结合构件动态性，分析构件迁移概率；利用路径分割的方式，分析不同路径的出现频率，并结合路径可靠性计算系统可靠性。3. 对于一些出现频率比较高的路径，由于在系统可靠性评价中的权重大，在实际系统开发中需要重点注意，在此路径中须采用可靠性高的构件方能提高系统可靠性。

结束语 本文采用了一种新的构件软件可靠性的计算方式，但模型的改进是以增加参数和计算复杂度为代价的，计算构件失效密度函数、确定各个构件在运行路径中的可靠度以及获取构件间的迁移概率都会增加一定的成本。

参 考 文 献

[1] Gokhale S S, Trivedi K S. Reliability prediction and sensitivity analysis based on software architecture[C]// IEEE 13th International Symposium on Software Reliability Engineering. 2002:64-75

[2] 毛晓光，邓勇进. 基于构件软件的可靠性通用模型[J]. 软件学报，2004，15(1)：27-32

[3] 邓勇进，毛晓光. 基于构件软件的可靠性通用模型及系统实现[J]. 计算机工程与科学，2005：27(3)：67-70

[4] Goseva-Popstojanova K, Hamill M. Architecture-based software reliability: Why only a few parameters matter[C]// The 31st Annual IEEE International Computer Software and Applications Conference(COMPSAC 2007). Beijing, 2007

[5] Katerina G, Trivedi K. Architecture-based approach to reliability assessment of software systems[J]. Performance Evaluation, 2001, 45(2): 179-204

[6] Delamarom. Integration testing using interface mutations[C]// The Seventh International Symposium on Software Reliability Engineering. New York, 1996: 46-52

[7] 刘宏伟. 非齐次柏松过程类软件可靠性增长型研究[D]. 哈尔滨:哈尔滨工业大学,2004

[8] 蔡开元,白成刚,钟小军. 构件软件系统的可靠型评估模型简介[J]. 西南交通大学学报,2003,37(6):551-554

[9] Kanoun K, Sabourin T. Software dependability of a telephone switching system[A]// The 17th International Symposium on Fault-Tolerant Computing[C]. Pittsburgh, Pennsylvania, 1987

[10] Gokhale S. An analytical approach to architecture based software relability prediction[A]// The Third International Computer Performance and Dependability Symposium[C]. Durham, North Carolina, 1998

[11] Everett W. Software componet reliability analysis[A]// The Symposium on Application-specific Systems and Software Engineering echnology[C]. Richardon, Texas, 1999

[12] 朱经纷,徐拾义. 软件可靠性综合模型的分析和研究[J]. 计算机科学,2009,36(4):181-184

[13] 陈俊文,谷建华,等. 一种改进的基于架构的软件可靠性模型[J]. 计算机工程与应用,2009,45(33):60-63

(上接第 143 页)

Web 环境下组合服务结构化演化方法 EM4CS,对 EM4CS 方法的流程进行了具体说明,并给出了遵循 EM4CS 方法的组合服务演化支撑系统 ESS4CS 的设计与实现。通过 EM4CS 方法的指导和 ESS4CS 系统的辅助支持,知识工程师能够快速地从问题域中抽取组合服务演化需求,并在机器辅助下对 OWL-S 服务描述进行修改,同时通过多次迭代处理完成服务描述语法和语义一致性检测与修复。在演化过程经过确认后,知识工程师能够将演化结果在受控的前提下发布,并使组合服务运行实例尽快同步这些变化,保持全局应用的一致性。在下一步工作中,我们将对 EM4CS 方法和 ESS4CS 系统进行进一步研究和完善,具体包括如下两个方面:

(1)本文所提出的 EM4CS 方法以采用 OWL-S 语言描述的组合 Web 服务为研究对象,未来需要进一步研究如何使 EM4CS 方法支持采用其他语言描述的组合服务,如 WSMO[17] 和 SWSF[18] 等;

(2)目前 ESS4CS 系统能够支持服务组合模式演化,但是无法实现服务运行实例依据演化结果在线调整。在未来的工作中,我们将进一步对支持演化的语义 Web 执行环境进行研究,使组合服务运行实例能够快速同步组合模式演化中所发生的变化。

参 考 文 献

[1] 喻坚,韩燕波. 面向服务的计算—原理和应用[M]. 北京:清华大学出版社,2006

[2] Haller A, Cimpian E, Mocan A, et al. WSMX-A Semantic Service-Oriented Architecture[C]//Proceedings of the IEEE International Conference on Web Services(ICWS 2005). IEEE Computer Society, Washington, DC, USA, 2005:321-328

[3] Norton B, Pedrinaci C, Domingue J, et al. Semantic Execution Environments for Semantics-Enabled SOA [J]. Information Technology, 2008, 50(2):118-121

[4] Ilkaev D. Recent Trends in Semantic SOA. 2007. [OL]. http://www.stickyminds.com/sitewide.asp? Function=edetail&ObjectType=ART&ObjectId=12612&tth=DYN&tt=siteemail&iDyn=2

[5] McIlraith S A, Son T C, Zeng Hong-lei. Semantic Web Services [J]. IEEE Intelligent Systems, 2001

[6] 刘飞. SOA 中国路线图. 2007. [OL]. http://www.ufida.com.cn/subject/200705xxhpl/pdf2/soa.pdf

[7] Karastoyanova D, Buchmann A. ReFFlow: a model and generic approach to flexibility of web service compositions[C]//iiWAS'2004 - The sixth International Conference on Information Integrationand Web-based Applications Services. Jakarta, Indonesia, September 2004

[8] Tao A T, Yang Jian. Supporting DifferentiatedServicesWith Configurable Business Processes[C]// IEEE International Conference on Web Services, 2007. 2007:1088-1095

[9] Naccache H, Gannod G C. A Self-Healing Framework for Web Services[C]// IEEE International Conference on Web Services (ICWS 2007). 2007:398-345

[10] Chafle G, Dasgupta K, Kumar A, et al. Adaptation in Web Service Composition and Execution[C]// Proceedings of the IEEE International Conference on Web Services(ICWS 2006). 2006:549-557

[11] Colombo M, Nitto E D, Mauri M. SCENE: A Service Composition Execution Environment Supporting Dynamic Changes Disciplined Through Rules[C]// Service-Oriented Computing ICSOC 2006. Springer: Berlin / Heidelberg, 2006:191-202

[12] Blanchet W, Stroulia E, Elio R. Supporting Adaptive Web-Service Orchestration with an Agent Conversation Framework[C]// Proceedings of the IEEE International Conference on Web Services(ICWS 2006). 2006:541-549

[13] Guarino N. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration[C]// International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology. 1997:139-170

[14] Kalfoglou Y, Hu B, Reynolds D, et al. Semantic integration technologies[C]// Proceedings of the 6th Month Deliverable. University of Southampton and HP Labs. ECS e-Prints Report # 10842, 2005

[15] Bao Ai-hua, Yao Li, Yuan Jin-ping, et al. Approach to the Formal Representation of OWL-S Ontology Maintenance Requirements[C]// the Proceedings of the Ninth International Conference on Web-Age Information Management 2008 (WAIM 2008). Zhangjiajie, Hunan, China, July 2008:56-61

[16] 鲍爱华,袁金平,姚莉,等. 基于扩展着色 Petri 网的 Web 服务本体描述语言过程语义分析方法[J]. 计算机集成制造系统,2008(09):1856-1864

[17] WSMO. Web Service Modeling Ontology (WSMO) - An Ontology for Semantic Web Services. 2005 6-9[OL]. http://www.w3.org/2005/04/FSWS/Submissions/1/wsmo_position_paper.html#omg

[18] Battle S, Bernstein A, Boley H, et al. Semantic Web Services Ontology (SWSO). 2005. [OL]. http://www.daml.org/services/swsf/1.0/swso/