ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.6, June 2008, pp.1350−1362
DOI: 10.3724/SP.J.1001.2008.01350

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种面向网构软件体系结构的信任驱动服务选取机制*

王 远[1,2+], 吕 建[1,2], 徐 锋[1,2], 张 林[1,2]

[1](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

[2](南京大学 计算机软件研究所,江苏 南京 210093)

An Internetware-Software-Architecture-Oriented Trust-Driven Mechanism for Selecting Services

WANG Yuan[1,2+], LÜ Jian[1,2], XU Feng[1,2], ZHANG Lin[1,2]

[1](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

[2](Institute of Computer Software, Nanjing University, Nanjing 210093, China)

+ Corresponding author: E-mail: wangyuan@ics.nju.edu.cn

Wang Y, Lü J, Xu F, Zhang L. An internetware-software-architecture-oriented trust-driven mechanism for selecting services. *Journal of Software*, 2008,19(6):1350−1362. http://www.jos.org.cn/1000-9825/19/1350.htm

Abstract: Based on the idea of trust evaluation, this paper proposes an Internetware software architecture oriented trust-driven mechanism for selecting services to solve the problem from the aspect of service selection: Firstly, a general, machine-readable description mechanism is proposed to express user requirements and trust evolution policies; secondly, a feed-back trust formation and decision mechanism is introduced and a trust-driven algorithm for selecting services is given; finally, an Internetware software architecture oriented trust-driven service selection framework is proposed for the development of trustable Internetwares. Some development examples indicate that this mechanism can support the development of the trustable Internetwares effectively.

Key words: Internetware; trust; trustworthiness; service; software architecture

摘 要: 基于信任度评估的思想,从软件服务选取的角度对此问题进行了探讨,提出一种面向网构软件体系结构的信任驱动的服务选取机制:首先,给出一种通用的、机器可理解的应用需求及信任演化策略描述规范;其次,采用一种基于反馈的信任形成及决策机制,并给出一个信任驱动的服务选取算法;最后,给出一种面向网构软件体系结构的信任驱动的服务选取支撑框架.初步实践表明,上述机制为开发可信网构软件提供了有效支持.

关键词: 网构软件;信任;可信性;服务;软件体系结构

中图法分类号: TP311 **文献标识码**: A

与传统运行环境不同,Internet 平台具有如下特征:无统一控制的“真”分布性、节点的高度自治性、节点链

* Supported by the National Basic Research Program of China under Grant No.2002CB312002 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant Nos.2006AA01Z159, 2007AA01Z140 (国家高技术研究发展计划(863)); the National Natural Science Foundation of China under Grant Nos.60233010, 60403014, 60603034 6060303 (国家自然科学基金); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2006712 (江苏省自然科学基金)

Received 2006-07-03; Accepted 2006-12-22

接的开放性与动态性、实体行为的不可预测性、运行环境的潜在不安全性、使用方式的个性化和灵活性等.Internet 使得计算机软件的运行环境由封闭、静态、可控逐步转变为开放、动态、多变.如何在 Internet 环境中集成和共享各类资源,成为当前软件技术所面临的重要挑战之一[1−3].基于 Internet 中的大量软件服务实体,出现了一类新型软件系统:此类系统以软件构件技术为支撑,通过自主、开放的服务实体的互连、互通、协作和联盟来构筑,称为网构软件[2,3].从宏观角度看,网构软件体系结构分为软件实体和协同方式[3],其中,软件实体以封装了各类资源的软件服务形式存在(称为软件服务),通常由第三方提供,具有独立、面向多用户的个体特征,以及量大面广、推陈出新的群体特征.协同方式是指在软件服务间建立联系、管理交互和协调活动的手段和方法[4].为了与软件服务的特征相适应,协同方式由简单的连线机制逐步演化为协同程序设计:允许软件服务采用多模式共存的方式进行交互,灵活、自主地对软件服务进行剪裁与组合,在满足个性化需求的同时,动态适应外部环境的变化.海量、多样的软件服务和灵活、自主的协同方式决定了网构软件体系结构具有开放、动态的特性.在该体系结构框架下,如何开发可信的网构软件成为当前的技术难点之一[1].

网构软件的可信性涉及安全性与可靠性等诸多侧面[1].开放、动态的体系结构使得常用的安全和可靠性技术具有一定的局限性:(1) 开放环境下对于海量、多样的软件服务缺乏绝对可信的第三方认证实体对其相关属性作出担保,传统的基于授权和认证的安全模型和技术不再适用于网构软件;(2) 对于许多具体系统而言(如基于 Web 服务的网上购物系统、查询系统、旅行代理系统等),通常只需获得“足够”可信的服务,传统安全方法过于严格,将限制系统的扩充和应用;(3) 软件服务通常由第三方提供,其服务质量难以保证,网构软件对其开发和运行过程无法进行直接管理,而常用的可靠性保障技术需依赖对软件系统的全局掌控;(4) 用户群体的开放性、多样性以及环境的复杂性,使得网构软件的开发不可能一蹴而就,其应该具有在运行时刻动态调整以适应需求的能力.因此在开放环境下,网构软件需要一种“柔性”的机制来保障其可信性.

本文工作的主要切入点是从网构软件的基本元素“软件服务”出发,借助信任评估[5,6]的思想来探讨此问题的解决途径:在协同方式的层次上,提出一种面向网构软件体系结构的信任驱动的服务选取机制(Internetware-software-architecture-oriented trust-driven mechanism for selecting service,简称 ISAOT).ISAOT 在协同方式上支持服务的动态绑定和选取,在服务的选取上以信任为基础进行决策.ISAOT 基于一类可动态连接软件服务的智能连接子——虚服务工作:1) 提供一个通用的基于 ontology 的应用需求及信任演化策略描述规范,用于解决多样化应用需求难以描述的问题,使得连接子可理解动态变化的应用需求,并触发信任演化行为;2) 利用开放协同环境中广泛使用的信任概念及其相关技术,采用一种基于反馈的信任形成及决策机制,并通过一个信任驱动的服务选取算法,促使连接子动态选取最为可信的服务,满足系统的动态需求;3) 以信任为通用的驱动机制,给出了一个面向网构软件体系结构的实现框架.ISAOT 将阐述网构软件随应用需求变化自主选取服务所需的相关原理、模型和方法,展示一个清晰的可信网构软件开发模型,并详细讨论其相关支撑设施的功能与实现机制.

本文第 1 节讨论信任相关概念.第 2 节详细论述 ISAOT 机制及其相关实现方法.第 3 节给出一个系统开发实例.最后讨论相关工作及总结.

1 信任相关概念

目前还缺少广泛、可接受的信任定义.不少学者认为,信任是一种主观的、情境相关、动态的概率变量,暗示了对相关实体未来行为的预测[5−7].在网构软件情境下,本文认为:

定义(信任). 信任是软件服务关于其他服务或服务集合在特定情境下具有完成某一特定任务能力可能性的主观判断,其程度依赖于服务对于信任对象的直接经验和推荐信息.

若服务 *A* 信任 *B*,则暗示 *A* 将对 *B* 在某种特定情境下完成某类操作 *C* 的能力进行预测.*A* 与 *B* 之间存在着信任关系.信任关系有不同的程度,本文利用区间(0,1)之间的实数对其进行量化,称为信任度.为使基于信任的分布式系统正常运行,合理量化和评价实体间的信任关系将是工作的重点.

信任关系与情境密切相关.情境(context)是一组关于特定情形的描述信息,该信息将当前情形与其他情形区分开来[8].在通常情况下,*A* 对于 *B* 的信任通常仅局限于某一特定的情境之内(如 *A* 信任 *B* 是一个好的商人并

不意味着 A 相信 B 是一个好的演员);特别是在允许推荐(recommendation)或基于声誉(reputation)的情况中,信任关系更加依赖于情境($(A\rightarrow_R B)\&\&(B\rightarrow C)\rightarrow(A\rightarrow C)$,其中,$\rightarrow_R$ 表示推荐信任关系,A 仅在 B 推荐的情境之内信任 C).在普适计算与智能系统中,情境通常采用基于 ontology 的方法描述,一些描述工具,如语义网络(semantic Web)、资源描述框架(resource description framework,简称 RDF)等被用来描述与情境相关的信息.在本文中,情境主要关注应用需求,通过判断协作者是否满足当前的应用需求来产生相关协作服务间的信任关系是本文的工作基础.

2 ISAOT:面向网构软件体系结构的信任驱动的服务选取机制

基于网构软件运行环境及其体系结构特征,网构软件需动态、自主地进行服务更替;网构软件应能理解应用需求,在满足功能的前提下选取合适的软件服务,以提升整个系统的效率与可信性.在给出信任驱动的服务选取机制之前,先来看一个网构软件系统实例:一个简单旅行代理系统.

如图 1 所示,在遵循调用标准的前提下,系统允许客户从不同的终端平台(PC,PDA 或 laptop)调用系统所提供的服务.该系统展示了网构软件体系结构所包括的 2 个因素:软件服务和协同方式(包括服务间交互、组合模式及其约束条件).软件服务包括系统所涉及的外部服务实体.服务描述及其交互可采用 Web 服务的相关技术实现(如 WSDL,SOAP 等).在进行服务组合时,系统引入了一类智能实体——虚服务,来替代传统体系结构中的连接子.所谓虚服务,是一类连接调用服务与被调用服务的特殊实体,其包含了客户端请求接口和为客户调用其他软件服务所提供的服务而留出的服务调用接口.虚服务允许软件服务在发布前不需要实现全部的接口,允许用户选取特定软件服务或服务组合来实现接口,增强了网构软件的灵活性与自治性.虚服务可提供软件服务动态组合的功能,为系统在运行时刻替换特定的软件服务提供支持.与传统软件体系结构中的连接子相比,虚服务是一类用于协同调用的智能实体,它应能感知应用需求,以信任驱动服务的动态替换与更新.虚服务需要约束条件加以限制.约束条件反映虚服务在软件服务调用及组合过程中所必须遵守的规范及要求,用于约束和指导虚服务的行为.图 1 所示的系统提供 2 种类型的虚服务:票务预定虚服务与酒店预定虚服务.每类虚服务用于连接多个外部软件服务与应用系统.

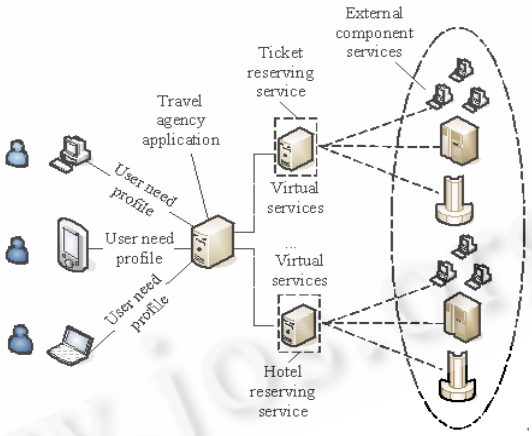


Fig.1 The example of Internetware: A travel-agency system

图 1 网构软件实例:一个旅行代理系统

基于虚服务的网构软件体系结构如图 2 所示.虚服务动态提供两类接口(客户端接口和服务调用端接口)连接服务请求者与服务提供者,并通过智能模块根据应用需求选取可信的服务.虚服务需要一个通用的规范来描述应用请求与返回结果的语义信息,并基于此评价请求与结果是否匹配、是否满足约束条件;评价结果将作为智能模块选取可信服务的依据.因此,ISAOT 的重点集中在如何使虚服务理解动态变化的应用需求及约束条件,以及如何促使虚服务自主的选取可信软件服务(智能性).本文将完成如下的主要工作:

• 提供一个通用的描述规范以描述虚服务两类接口所需的语义信息,并基于此规范定义相关信任演化策略,使得虚服务依策略更新相关服务信任值成为可能.

• 提供一个合理的信任形成及决策机制,并给出一个基于信任的服务选取算法.虚服务的智能模块通过执行该算法,合理选择软件服务,尽最大可能来提高用户满意度.

• 提供一个信任驱动的可复用、支持可信软件服务选取的实现框架.该框架应从方法学的角度及程序设计的层次为可信网构软件的开发提供支持.

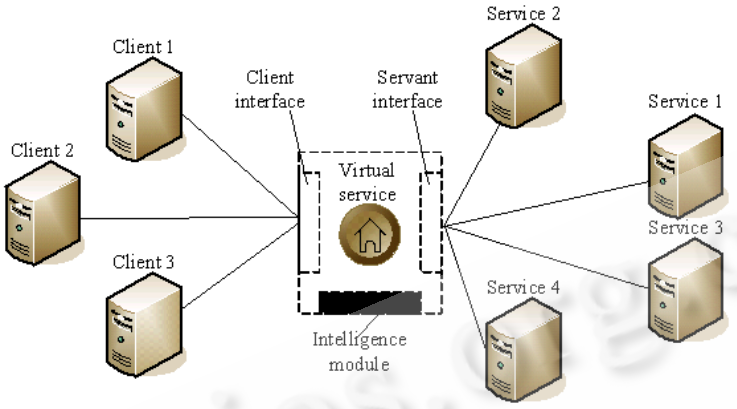


Fig.2 Virtual-Service-Based software architecture

图 2 基于虚服务的网构软件体系结构

2.1 情境相关的应用需求及信任演化策略描述规范

应用需求是情境的一种,本节借鉴文献[8]中的相关工作,提供一种基于 ontology 的应用需求描述规范,称为服务情境描述语言(service context description language,简称 SCDL).图 3 显示了 SCDL 的基本结构.

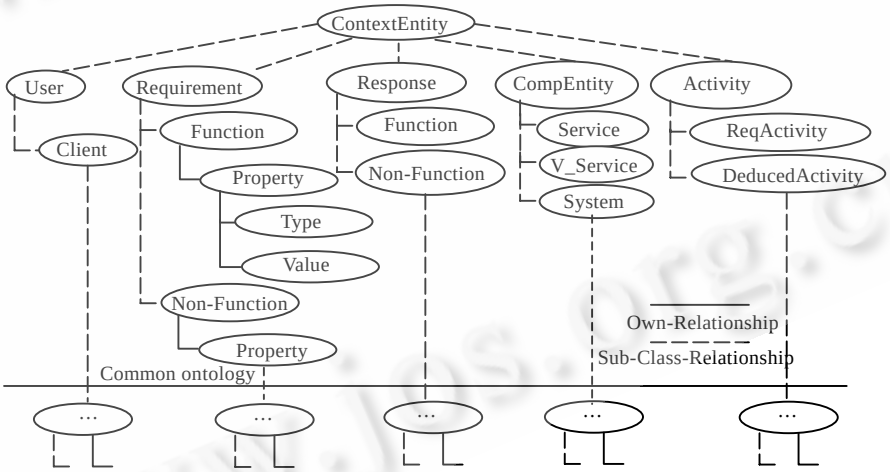


Fig.3 Ontology-Based context definition of SCDL

图 3 SCDL 中基于 ontology 的情境定义

SCDL:是 RDF 和 OWL 的一个扩展,可支持知识共享、知识复用等功能.结合一定的逻辑推导机制,可完成从低层信息向高层信息的转化.在开放协同环境中,对服务的信任度取决于对其行为及相关细节的观测.对于虚服务而言,与需求和结果相关的数据及信息属于低层信息.虚服务形成信任关系并触发选取行为,依赖于对这些信息的认知.SCDL 能够以一种机器可理解的方式描述应用需求与返回结果.SCDL 采用基于面向对象的方法:情境中的每一个实体都是 ContextEntity 实体的子类.SCDL 定义了 5 个直接继承于 ContextEntity 的子实体:用

户(user)、需求(requirement)、响应(response)、计算实体(CompEntity)和行为(activity).

用户:指调用网构软件所提供服务的人或客户端软件.用户定义子类 Client 指定特定的客户端软件.

需求:表明了用户期望获得的服务调用结果的一些重要属性,用于客户端接口语义信息描述.在传统的软件工程中,需求分为 2 大类:功能性需求(function)与非功能性需求(non-function).通常情况下,功能性需求刻画了系统提供的服务所必须达到的标准,必须严格被满足;非功能性需求则主要刻画了系统性能及系统维护相关的特性,包括可靠性、安全性、低延迟和可维护性等.在 SCDL 中,非功能性需求主要用于表达用户对系统安全、性能及可用性方面的要求.需求可以具有属性(property),由数值和类型组成.需求可用于表达虚服务所必须满足的约束条件.

响应:对应于特定的需求,用于描述返回的结果,具有与需求相同的语法和结构,用于服务调用端接口语义信息的描述.

计算实体:指特定的网构软件系统、Internet 中的各种软件服务和虚服务.SCDL 为计算实体定义了 3 个子类:Service,V_Service 和 System.Service 用于表示软件服务,V_Service 指代虚服务,System 指代特定的系统.

行为:指 ContextEntity 的一系列的动作或关系.行为分为两类:一类指代用户请求,由 ReqActivity 表示;另一类为策略中 ContextEntity 间的关系或动作,称为 DeducedActivity.

结合图 1 旅行代理系统的例子,给出 SCDL 情境定义的一个简单应用.以下是旅行代理系统所收到的一个用户请求:Mary 预定 2 张航班 UA858 的经济舱,且其优先级为高.

图 4 给出了该请求所涉及 ContextEntity 的 SCDL 描述.

```
〈SCDL: Class rdf: ID=“TicketClass”〉
    〈rdfs: subClassOf rdf: resource=“#ContextEntity”〉
    〈SCDL: enum SCDL: Domain=“First,Business, economy”〉
〈/SCDL: Class〉
〈SCDL: Class rdf: ID=“FlightInfo”〉
  〈rdfs: subClassOf rdf: resource=“#Function”/〉
  〈SCDL: Property rdf: ID=“N.O.”〉
    〈SCDL: type rdf: resource=“xsl: string”/〉
    〈SCDL: value rdf: resource=“UA858”/〉
  〈/SCDL: Property〉
  〈SCDL: Property rdf: ID=“Amount”〉
    〈SCDL: type rdf: resource=“xsl: int”/〉
    〈SCDL: value rdf: resource=“2”〉
  〈/SCDL: Property〉
  〈SCDL: Property rdf: ID=“Class”〉
    〈SCDL: type rdf: resource=“SCDL: TicketClass”/〉
    〈SCDL: value rdf: resource=“economy@TicketClass”/〉
  〈/SCDL: Property〉
〈/SCDL: Class〉
```

(a)

```
〈SCDL: Class rdf: ID=“ReserveTicket”〉
  〈rdfs: subClassOf rdf: resource=“#ReqActivity”/〉
〈/SCDL: Class〉
```

(b)

```
〈SCDL: Class rdf: ID=“Mary”〉
  〈rdfs: subClassOf rdf: resource=“#User”/〉
〈/SCDL: Class〉
```

(c)

```
〈SCDL: Class rdf: ID=“Urgency”〉
  〈rdfs: subClassOf rdf: resource=“#Non-Function”/〉
  〈SCDL: Property rdf: ID=“delay”〉
    〈SCDL: type rdf: resource=“xsl:double”/〉
    〈SCDL: value rdf: resource=“10s”/〉
    〈/SCDL: Property〉
〈/SCDL: Class〉
```

(d)

Fig.4 The example of ContextEntity

图 4 ContextEntity 应用实例

请求可以描述为:Mary ReserveTciket with FlightInfo, Urgency.该请求同时也可看作一类特定情境下的虚服务约束条件,当该请求不被满足时,意味着服务实体不能满足需求,虚服务需要选取其他合适的外部服务.SCDL 允许自定义所需的 ContextEntity.如图 4(a)中的 TicketClass,此实体用于标注舱位的等级.对于返回结果(response),SCDL 可采用相同的方法进行描述.SCDL 语法基于标准的 XML 语言,有利于与现存标准(如 RDF,WSDL,SOAP)的结合.

SCDL 可支持基于一阶谓词逻辑的信任演化策略定义.继续考虑图 1 的例子,图 5 定义了旅行代理系统中的信任更新策略.旅行代理系统定义了 3 个 DeducedActivity:

• *Satisfy*:二元关系,用于推断返回结果(response)是否与用户需求(requirement)相符.

• *Provide*:二元关系,用于表达组件服务(service)返回结果(response)的行为.

• *Update*:二元关系,用于表达虚服务(V_service)更新信任值的行为.

```
〈SCDL: Class rdf: ID="Satisfy"〉
  〈rdfs: subClassOf rdf: resource="#DeducedActivity"/〉
  〈SCDL: domain rdf: resource="#Requirement"/〉
  〈SCDL: range rdf: resource="#Response"/〉
〈/SCDL: Class〉
```

```
〈SCDL: Class rdf: ID="provide"〉
  〈rdfs: subClassOf rdf: resource="#DeducedActivity"/〉
  〈SCDL: domain rdf: resource="#Service"/〉
  〈SCDL: range rdf: resource="#Response"/〉
〈/SCDL: Class〉
```

```
〈SCDL: Class rdf: ID="Update"〉
  〈rdfs: subClassOf rdf: resource="#DeducedActivity"/〉
  〈SCDL: domain rdf: resource="#V_Service"/〉
  〈SCDL: range rdf: resource="xml: double"/〉
〈/SCDL: Class〉
```

$$\forall u,u\in User,\exists s,vs,s\in Service,vs\in V\_Service,$$
$$Provide(s,Response)$$
$$\&\&\ Satisfy(Response,u.requirement)$$
$$\rightarrow Update(vs, tv^{vs\rightarrow s}_{u.requirement})$$

Fig.5 The definition of DeducedActivity-Based trust update policy

图 5 基于 DeducedActivity 的信任更新策略定义

在图 5 中,$tv^{vs\rightarrow s}_{u.requirement}$ 表示在情境 *u.eequirement* 下,*vs* 对于 *s* 的信任值.策略含义为:当服务提供的结果能满足用户需求时,提高服务在该情境下的信任度.相应地,我们可定义降低服务信任度的策略.

SCDL 可给出 DeducedActivity 的定义,但其实现还依赖于具体的系统.以图 5 中的策略为例,*Provide* 的语义与实现均较为简单.*update* 的实现涉及到分布式环境下信任数据收集,信任的建模以及信任模型的应用将在下一节详细讨论.*Satisfy* 用于检验响应是否符合的请求,其必须对所定义的请求中的每一个 Property 进行检验.递归的检验算法描述如下:

```
boolean check (Requirement req, Response res)
  boolean cons=false;
  for each property e in req do          //依次检测请求中的每一个属性 e
    select e′ from res;                  //从响应(response)中选取对应的属性 e′
      cons=checkproperty(e,e′);          //检测 e′与 e 是否匹配
    if (!cons) break;
  return cons;
boolean checkproperty (Property p1, Property p2)
  boolean cons=false;
  if !(p1 and p2 are both basic type or structure)    //不属于同一结构类型,返回 false
        return cons;
  if (p1 and p2 are both basic type)                  //属于基本结构类型时,检测是否相同
      cons=checkbasic(p1,p2);
  else                                                //对于复杂数据类型,依次检测其子数据项
    for each subItem in p1
      select subItem′ in p2
      cons=checkproperty(subItem,subItem′);
      if !cons
          break;
  return cons;
```

check 函数的实现可利用聚类算法[9]中的数据规范化(normalization)技术、矢量空间模型(VSM)、相异度(dissimilarity)模型等成熟技术.SCDL 允许用户为每一个 Property 自定义检验方法,通过标签 SCDL:check 和 SCDL: checkMethod 标注,其语法如下:

```
…
〈SCDL: Property rdf: ID="N.O."〉
    〈SCDL: type rdf: resource="…"/〉
    〈SCDL: value rdf: resource="…"/〉
    〈SCDL: check SCDL: checkMethod="com.agency.checkNO"/〉
〈/SCDL: Property〉
```

2.2 基于反馈的信任形成及决策机制

基于 *satisfy* 操作结果,虚服务可更新相应的信任值(*update* 操作).信任值的准确性决定了服务选取的效果.我们在文献[10−12]中对信任度的计算进行了研究,本节将其中一些结论应用于网构软件开发.本节所采用的信任更新方法以推荐信息和自身经验为依据.当信任值与经验及推荐信息不符时,系统则对相应的信任值进行调整,在计算新信任值时,以当前信任值为输入,故称为基于反馈的信任形成机制.经验与推荐信息分别定义如下:

定义(经验). 经验为一个三元组:⟨*Object*,*Requirement*,*Record*⟩,其中,*object* 为开放协同环境中特定的软件服务(*Service*);*Requirement* 为调用 *Object* 时用 SCDL 规范所定义的应用需求;*Record* 为 *Object* 的调用记录,为一个二元组⟨*s*,*a*⟩,其中,*s* 表示成功调用次数,*a* 表示总的调用次数.经验记录了软件服务在特定需求下的运行情况:当 *satisfy* 返回真时,记为成功经验;否则记为失败经验.

定义(推荐信息). 推荐信息为一个二元组:⟨*Rec*,*Exp*⟩,其中,*Rec* 表示推荐者,为开放环境中具有推荐能力的计算实体(*CompEntity*).*Exp* 为 *Rec* 所提供的经验信息,*Rec* 为 *Exp* 的所有者.

利用聚类算法[9]可以将收集到的推荐信息根据用户给定的 *Requirement* 进行分类.分类的依据是语义距离.语义距离是衡量不同用户需求之间差异的标量,其定义如下:

定义(语义距离). 对于任意两类不同需求 $Req_1=\langle p_1,p_2,\ldots,p_i\rangle$,$Req_2=\langle p_{i+1},p_{i+2},\ldots,p_n\rangle$(p 为所定义的属性),则语义距离(标记为 d)为:$d=1-same(Req_1,Req_2)/diff(n)$,其中,$same(Req_1,Req_2)$表示两类需求中相同的属性个数;$diff(n)$表示 n 个属性中相异的属性个数.

设定一个阈值 *T*,对于任意语义距离小于 *T* 的需求均可视为同类需求,由此得到一个与给定 *Requirement* 相关的推荐信息集合 ω,结合自身的经验信息 *e*,生成关于服务提供组件 *P* 的综合经验 $c=\langle P,Requirement,r\rangle$,*r* 的计算方法如下:

$$r.s=\sum_{I\in\omega} I.Exp.s\times t_I+e.s;\ r.a=\sum_{I\in\omega} I.Exp.a\times t_I+e.a.$$

t_I 为系统对于每一个推荐者的推荐信任度.关于推荐信任度的计算,在文献[11]中有详细的论述.根据生成的综合经验 *c*,按如下步骤更新当前的信任值:

(1) 信任度暗示了相关软件服务未来操作成功的概率.利用假设检验的思想,判断当前的信任度是否与综合经验吻合,称为信任评估.其算法如下:

(1.1) 设定假设 H_0:当前信任度 *t* 与综合经验 *c* 中的 *r* 所记录的推荐信任度 t'($t'=r.a/r.s$)相匹配;选取合适的显著水平 α.

(1.2) 中心极限定理[13]设定统计量 $U=\dfrac{r.a\times|(r.s/r.a)-t|}{\sqrt{r.a\times t\times(1-t)}}$,随着 *r* 的递增,*U* 服从标准正态分布.

(1.3) 根据公式 $P\{U\geq k|H_0\}=\alpha$,得 $k=\int_{-\infty}^{1-\frac{\alpha}{2}}\dfrac{e^{-x^2/2}}{\sqrt{2\pi}}dx$.若 $U\geq K$,则不接受 H_0,反之则接受.

(2) 若 H_0 成立,则保持当前信任度不变;若 H_0 不成立,则依据下面的式(1)更新信任度.

$$U(t,r)=\begin{cases}t+\dfrac{-In(t)}{r.a+In(t)}p(1-t), & p>t\\ t, & p=t\\ t-\dfrac{-In(1-t)}{r.a+In(1-t)}(1-p)t, & p<t\end{cases}\tag{1}$$

其中,$p=r.s/r.a$.

式(1)是文献[10]中的信任演化公式在本文中的具体应用.该公式基于反馈模式,配合信任评估过程,能根据信任相关信息(综合经验 *c*),从当前任意的信任度进行迭代,一直得到较为准确的信任度为止.使用该公式可以将信任的生成与更新操作合并.系统可任意设定初始值而不影响最终结果的生成.公式利用了 Shanon 信息论中的相关理论,利用当前信任度所蕴涵的信息量($-In(t)$),结合相关实体之间的经验值,计算信任度.

基于该信任公式,给出信任驱动的服务选取算法进行决策,由虚服务的智能模块执行.算法分为两个阶段:

初始化阶段和运行阶段.初始化阶段为每个待选软件服务分配一个信任初值,可为任意位于区间(0,1)的实数.初始阶段仅在系统部署的时候运行一次.运行阶段在每次接收到服务请求时调用,以选择信任度最高的软件服务参与系统运行.该算法在最坏情况下的时间复杂度为 $O(n\times m)$(n 为待选软件服务的数量,m 为应用需求中的属性个数),但是,一次调用成功率的提高可以大大提高算法效率.信任值的准确程度将决定一次调用成功率的高低,因此,算法的实际性能依赖于信任形成和更新机制.第 3 节将通过应用实例分析上述信任形成和更新机制及服务选取算法在实际应用中的有效性.

```
Initialization stage:
    for each service in the application        //为每一个代选服务配置候选者列表
        Construste(Candidate_list);
    for each item in the candidate_list        //为每一个候选者设定初始信任值
        item.trustvalue=init_value;
Running stage:
    wait until a user requirement Req arrive;
    service_type=Req.purpose;
    is_disired_result=false;
    do {
    //选取可信度最高的组件进行调用,若多个候选者可信度相同,则随机选取一个
        component=SelectReliableComp(service_type);
        Res=component.invoke();
        if (check(Req,Res)==false){
    //将失败组件移入失败列表,失败列表中的组件在当前服务中将不再被调用
            Remove the component to the fail_list;
        } else
            is_desired_result=true;
        update the experience in the related entry of the collector table; //更新经验
        if (!consistent(trust_value,exp)==true)                    //检测当前信任值与经验是否相符
            trust_value=U(trust_value,exp);                        //更新信任值
    } until ((is_desired_result) or (candidate_list equals fail_list));
        if (is_desired_result)
            return Res;
        else
            report exception and apply the people interference;
```

2.3 基于设计模式的信任驱动的ISAOT结构框架

本节面向网构软件体系结构给出一个基于设计模式的信任驱动的服务选取框架,用于指导将 ISAOT 机制应用于网构软件开发.框架基于截获者设计模式(interceptor)与过滤器设计模式(filter),规范开发过程中所涉及的原理、技术和方法,使系统集成者可以脱离实现细节而关注于应用逻辑.框架综合考虑了用户与系统、系统与组件之间的交互技术、不同组件服务间的协同技术、用户需求描述技术以及相关的信任建模计算方法.其核心组成部分如图 6 所示,包括:用户视图、拦截器、系统模块、元数据库、软件服务及交互协议等.

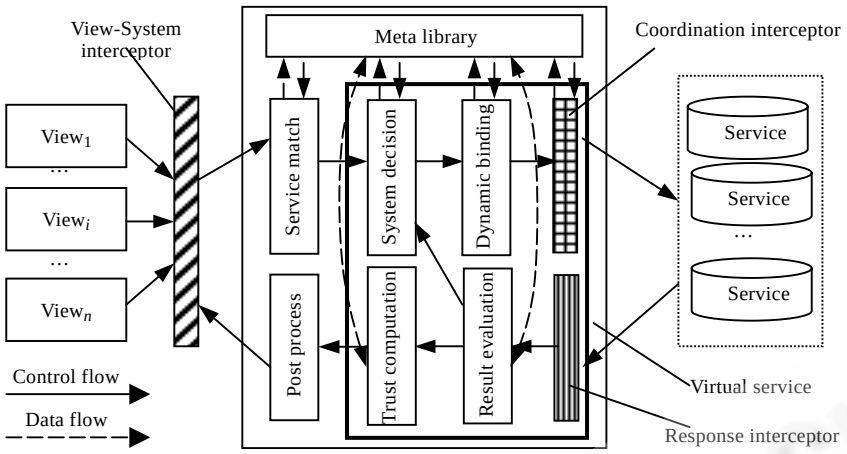


Fig.6 Pattern-Based and trust-driven architecture for ISAOT

图 6 基于设计模式的信任驱动的 ISAOT 结构框架

用户视图:提供模板供特定的用户群体输入需求.网构软件运行于 Internet 中,其用户群体开放多样.基于 ontology 的描述方式虽然通用,但是也需要一定的学习曲线,并不适用于所有的用户,特别是不具备计算机及网络相关知识的大量普通用户.系统提供者通过为不同的用户定制不同的视图,为用户提供可视化界面来解决这一问题.

视图-系统拦截器:将由用户视图获得的用户输入转化为 SCDL 格式.视图与拦截器之间需要特定的传输协议,视图提供者需要遵循此协议.视图-系统拦截器使得系统开发者关注于系统逻辑实现.

元数据库:保存各种基于 SCDL 的 ContextEntity 定义、服务接口定义、策略定义,以及信任相关信息(经验、信任值、推荐信任度等).开放环境下缺乏集中可控的机制,本地的信任相关信息不足会导致信任评估过程的不准确,元数据库应具有与其他同等元数据库交互的功能以收集信任相关信息.元数据库关于信任相关信息维护的 3 张表是:1) 信任度表:用户需求 context、服务 ID、信任值;2) 经验信息表:用户需求 context、服务 ID、提供者 ID、经验信息(⟨成功次数,总次数⟩);3) 推荐者表:用户需求 context、推荐者 ID、信任值.信任度表用于系统决策.经验信息表根据特定的交互协议定期更新.推荐者表由集成方在部署系统时配置,但是随着时间的进展及自身经验的增长,推荐信任值应相应地进行调整.文献[11]中讨论了推荐信任度的计算方法.

服务匹配模块:据 SCDL 所描述的用户需求,匹配一组与用户需求语义相近的服务.

系统决策模块:根据每个软件服务在特定用户需求(context)下的信任值选取合适的组件服务,完成基于信任的系统决策过程.

动态绑定模块:提供动态绑定实体的功能,使得系统可以在运行期选取合适的服务实体,则该次请求系统将与选定的软件服务协作完成.

协同调用拦截器:按照软件服务的接口规范,从用户需求中提取相应的属性(properties),组织成调用参数(interface signatures).

软件服务:遵循一定规范的、可独立开发、部署、运行的软件模块,包括各类 Web 服务、EJB 组件等.

协同响应拦截器:将服务返回的结果按照 SCDL 规范重新组织,以利于系统进行评估.

结果评价模块:利用第 2.2 节的检验算法评价返回结果.

信任计算模块:根据协同结果重新计算相应软件服务在特定 context(用户需求)下的信任值,并更新元数据库中的相应的数据.

后处理模块:该模块为可选模块,用于系统对于返回结果进行后处理.

协议:规定了视图与系统、系统与外部软件服务、系统模块与元数据库系统以及不同应用系统元数据库之

间的交互规范.

框架给出了自适应可信网构软件结构及其支撑设施,其中,系统决策模块、动态绑定模块、协同调用拦截器、协同响应拦截器、结果评价模块和信任计算模块包含于虚服务构件中.图 6 中的控制流标记指明了各模块协同工作的流程.ISAOT 框架中,软件开发者本质上是一个软件服务集成者,所需完成的工作包括:

1) 分析用户群体,提供相应的用户视图.此项工作亦可交由用户完成,用户可以在遵循视图与系统之间交互协议的前提下定制自己的视图,然后将本视图部署到相应的服务器上.视图与系统将通过 B/S 模式或相关调用模式(如 RPC 等)进行交互.

2) 建立并维护元数据库.系统部署之前,建立元数据库所需的 3 张表格,按照元数据库之间的协议,给定元数据库经验信息及推荐者信息的更新策略.

3) 建立虚服务,根据虚服务语义为其选取一组预定的软件服务作为系统运行时的候选者,并指定虚服务的调用次序(应用逻辑).

4) 为特定的用户需求中的各种属性提供特定 checkMethod 以检测其相关性.

5) 为系统提供必要的后处理方法.

3 系统实现实例分析:一个网络订票代理系统

基于 ISAOT,本文实现了图 1 所示的旅行代理系统中的订票服务功能:在 3 台 PC(P4 1.8G,512M 内存,WindowsXP 系统,Tomcat4.2 应用服务器,100M 以太网卡连接的局域网)机上部署了 5 个不同的虚拟订票软件服务(命名为服务 1~服务 5),其中,服务 1、服务 2 位于一台机器上,服务 3、服务 4 位于另一台机器上,服务 5 单独位于一台机器之上.通过策略(如网络负载均衡策略、CPU 负载均衡策略等)约束,5 个软件服务的调用成功率(调用成功率均为近似值)是:服务 1 为 10%,服务 2 为 30%,服务 3 为 50%,服务 4 为 70%,服务 5 为 90%.同时,设定 10 个推荐者,并将其分为 3 组:1) 3 个提供真实的推荐信息;2) 3 个提供恶意推荐信息;3) 4 个随机给出推荐信息.所有推荐者的初始推荐信任度均设为 0.5,并按照文献[11]中的方法更新推荐者的信任度.

系统的工作流程如图 7 所示.系统首先接受用户输入进行分析,选取适当的虚服务;随后,虚服务选取适当的软件服务以供调用;当获得结果后,分析结果是否符合用户需求,并更新软件服务的相关信任数据;最后将结果返回给用户.其中,应用需求(包括功能性需求与非功能性需求)可看作对虚服务的约束条件,虚服务需根据用户需求选取合适的服务组件,服务组件所返回的结果需符合约束条件,否则,虚服务将对软件服务进行替换.

在我们调用不同的软件服务的同时,通过式(1)更新每个软件服务在特定时段对应的信任值,以检验信任更新公式的有效性,并通过比较随机选取服务实体的调用模式与 ISAOT 调用模式在一次调用成功率上的差异来分析信任驱动服务选取算法的性能.

图 8 给出了一组有代表性的实验统计结果(共进行了 10 组模拟实验,每次交互次数为 100 次).在实验的初始阶段,ISAOT 由于缺乏历史经验信息,所以不能稳定地选取服务实体,因此,曲线较为曲折.这一点与随机选取方式较为相似.随着交互次数的增多,ISAOT 能够显著地提高一次交互成功率,并将其维护在一个较高的水准;而随机选取方式一次调用成功率随着交互次数的增加而大大低于 ISAOT.图 8(a)显示了每个不同的软件服务所对应的信任值,其大小排序与对应软件服务固有的计算能力相吻合;图 8(b)指明在基于信任的服务选取策略下,系统的一次调用成功率将远远高于随机调用模式;图 8(c)显示了实验过程中每一类推荐者的平均信任度.通过比较图 8(a)中服务 5 的信任曲线与图 8(b)中基于信任选取的一次调用成功率曲线可以发现,最可信实体的信任度与一次调用成功率基本相同;一次调用成功率的提高可以大大提高算法性能.结合算法可知,通过式(1)计算的软件服务可信度与软件服务的计算能力相一致,表明了利用该服务选取机制可使服务的选取达到最优.订票服务系统的相关数据验证了 ISAOT 机制的有效性.该机制在系统的运行初期需要一定的学习时间以获得实体的信任相关信息,在本例中,此段时间约为 8~10 次服务调用.在信任关系形成后,该机制可以最大限度地提高一次调用成功率,从而提升系统性能.实体最终信任度的形成与信任度初值无关,仅依赖于交互的次数与交互的结果,具有一定的通用性.

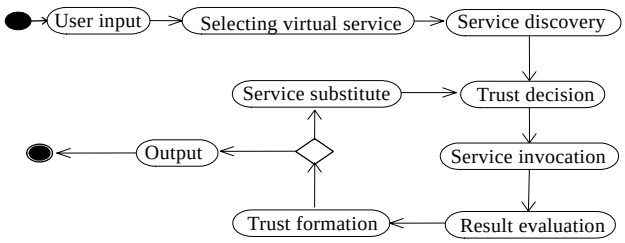
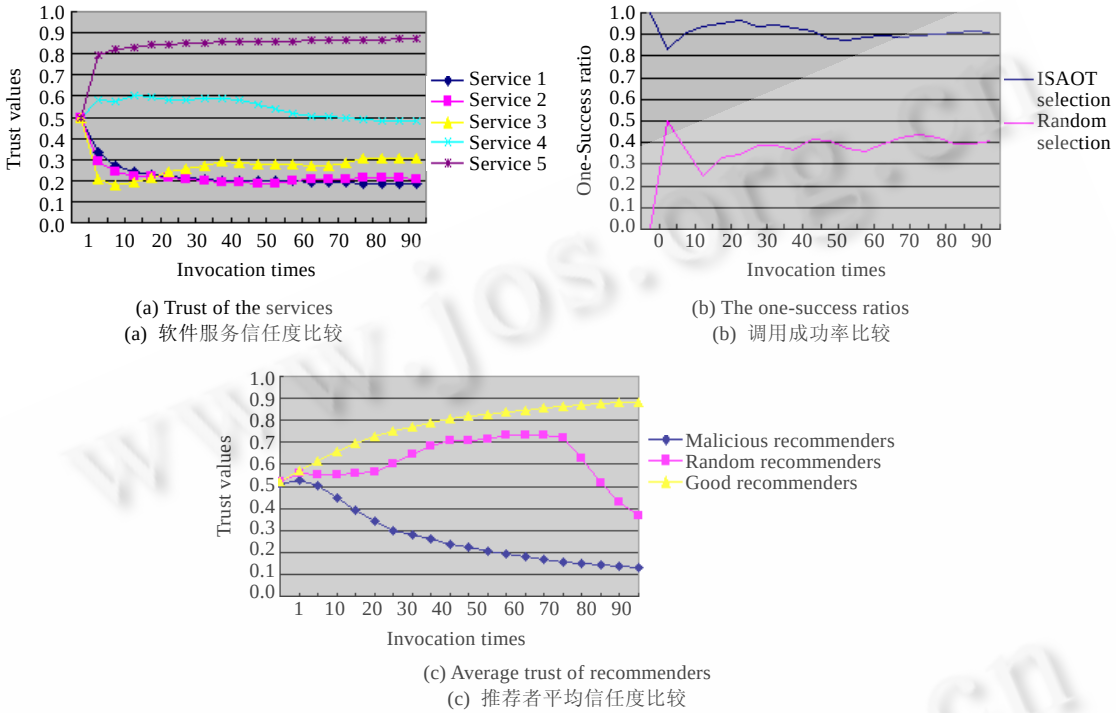


Fig.7 The activity diagram of service selection

图 7 服务选择过程活动图



(a) Trust of the services

(a) 软件服务信任度比较

(b) The one-success ratios

(b) 调用成功率比较

(c) Average trust of recommenders

(c) 推荐者平均信任度比较

Fig.8 Experiment results

图 8 仿真实验结果

通过构造上述订票系统,ISAOT 体现了其在服务构件选取方面的优势:合理利用历史经验信息以支持系统决策.ISAOT 适用于一类需求多变、可信需求相对“软化”的应用系统,称为日用系统(everyday software),而对于安全级别较高的系统并不适用.日用系统主要用于解决日常事务,其开发过程要求快速、开发费用要求节省.同时,其应具有一定的灵活性,当需求或环境发生变化时,可及时作出调整.网构软件是构造此类软件系统的一类有效编程模型,ISAOT 从服务选取的角度为网构软件的开发提供了支持.

4 相关工作及总结

相关工作涉及以下几个方面:在程序设计理念方面,美国 CMU 大学的 Shaw 等人提出“开放软件联盟”的思想[14],倡导以“足够正确”的方式构建系统,提出系统应具有“内稳态”的属性.本文可视作此理念下的一种编程模型及软件实现机制——给出了具体的服务(或资源)选取算法并讨论了其系统结构及支撑机制;在体系结构及演

化技术方面,Garlan 等人提出的 Rainbow,以及 Magee 等人提出的 Darwin 等[15,16]在交互协同模式的结构化方面作做了一定的工作.本文则主要在协同方式层次上,探讨了一种基于信任的服务选取机制,适用于服务的选取与替换,以服务替换的模式支持系统的演化,具有一定的通用性,可适用于不同的情境;在信任建模与计算方面,Abdul- Rahman 提出了一个简单的分布式信任模型以支持信任信息的有条件传递,并给出了相关数学处理方法[5];Beth 模型则给出了基于历史信息形成可信度的方法,其局限在于仅能利用肯定信息[6];Jøsang 模型[7]引入了“证据空间”和“观念空间”的概念,并采用主观逻辑算子给出了信任度的推导和综合计算方法,其局限性在于过于复杂.这些模型讨论了分布式环境下信任的建模与计算技术,但在适用性、有效性和可操作性方面存在一定的局限性,不适用于网构软件的开发.结合网构软件动态开放的特点,本文在相关工作[10−12]的基础上,给出了一套适用的信任形成与决策技术,以此作为网构软件演化的“驱动力”.

面向网构软件体系结构的信任驱动的服务选取机制 ISAOT 在体系结构模型、基础支撑设施、信任建模与计算以及支持系统运行时刻动态更新服务等方面做了一定的工作,在一定程度上解决了软件服务难于选取、其可信性难于保证的问题.主要工作包括:(1) 提出一种以虚服务连接的网构软件体系结构,该结构为支持软件服务动态选取与更替奠定了基础;(2) 针对网构软件用户群体的开放性和多样性,提出了一种基于 ontology 的应用需求描述规范 SCDL.在功能上,SCDL 使得虚服务可理解动态变化的应用需求;在语法上,SCDL 是 RDF 的一个扩展,与相关 XML 技术保持了较好的兼容性;(3) 利用 SCDL 所定义的 ContextEntity,基于一阶谓词逻辑给出了信任演化策略描述机制;(4) 给出一个有效的信任形成及决策机制和一个信任驱动的服务选取算法;结合 SCDL 及信任演化策略,服务选取算法可使得虚服务具有感知应用需求动态选取可信服务的能力,实现了虚服务的“智能”化;(5) 给出一个基于设计模式的信任驱动的实现框架,从程序设计的角度为可信网构软件的开发提供了支持;(6) 通过具体的开发实例,初步验证了 ISAOT 的有效性.ISAOT 使得可信软件服务的选取有了相应的模型及方法支持,初步解决了在开放协同环境中可信服务的选取问题,为可信网构软件的开发提供了支持.进一步的工作包括:如何高效、准确地收集相关信任信息;如何在替换之后确保虚服务在语义功能上的正确性;提供一批可视化编程工具,以支持基于 ISAOT 的网构软件开发;将本文工作进一步应用于大型系统的开发等.

References:

[1] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm

[2] Yang FQ, Mei H, Lü J, Jin Z. Some discussion on the development of software technology. Acta Electronica Sinica, 2002,30(12A):1901−1906 (in Chinese with English abstract).

[3] Lü J, Tao XP, Ma XX, Hu H, Xu F, Cao C. Research on Agent-based Internetware. Science in China (Series E), 2005,35(12):1−21 (in Chinese with English abstract).

[4] Ma XX. Research on software coordination on internet [Ph.D. Thesis]. Nanjing: Nanjing University, 2003 (in Chinese with English abstract).

[5] Abdul-Rahman A, Hailes S. A distributed trust model. In: Proc. of the 1997 New Security Paradigms Workshop. Cumbria: ACM, 1997. 48−60.

[6] Beth T, Borcherding M, Klein B. Valuation of trust in open network. In: Proc. of the European Symp. on Research in Security. Brighton: Springer-Verlag, 1994. 3−18.

[7] Gambetta D. Can we trust trust? In: Gambetta D, ed. Trust: Making and breaking cooperative relations. Basil Blackwell: Oxford Press, 1990. 213−237.

[8] Gu T, Wang XH, Pung HK, Zhang DQ. An ontology-based context model in intelligent environments. In: Proc. of the Communication Networks and Distributed Systems Modeling and Simulation Conf., Soc. for Modeling and Simulation Int’l. 2004.

[9] Han JW, Kamber M. Data Mining: Concepts and Techniques. Academic Press, 2000.

[10] Wang Y, Xu F, Lü J, Zhang L. A trust-based approach to select reliable components for Internet coalition applications. In: Proc. of the 9th IASTED Int’l Conf. on Software Engineering and Applications. 2005. 134−139.

[11] Wang Y, Xu F, Lü J. Establishing recommendation trust relationships for Internetwares. ACM SIGSOFT Software Engineering Notes, 2006,31(1):1−5.

[12] Wang Y, Lü J, Xu F, Zhang L. A trust measurement and evolution model for Internetwares. Journal of Software, 2006,17(4):682−690 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/682.htm

[13] Gao ZX, Chen HJ. Probability and Statistics. Nanjing: Nanjing University Press, 1995 (in Chinese).

[14] Shaw M, DeLine R, Klein DV, Ross TL, Young DM, Zelesnik G. Abstraction for software architecture and tools support them. IEEE Trans. on Software Engineering, 1995. 314−335.

[15] Galan D, Cheng SW, Huang AC, Schmerl B, Steenkiste P. Rainbow: Architecture-Based self-adaptation with reusable infrastructure. IEEE Computer, 2004,37(10):46−54.

[16] Magge J, Dulay N, Kramer J. Regis: A constructive development environment for distributed programs. Distributed System Engineering Journal, 1994,1(5):304−312.

附中文参考文献:

[1] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm

[2] 杨芙清,梅宏,吕建,金芝.浅论软件技术发展.电子学报,2002,30(12A):1901−1906.

[3] 吕建,陶先平,马晓星,胡昊,徐锋,曹春.基于 Agent 的网构软件模型研究.中国科学(E 辑),2005,35(12):1−21.

[4] 马晓星.Internet 软件协同技术研究[博士学位论文].南京:南京大学,2003.

[12] 王远,吕建,徐锋,张林.一个适用于网构软件的信任度量及演化模型.软件学报,2006,17(4):682−690. http://www.jos.org.cn/1000-9825/17/682.htm

[13] 高祖新,陈华钧.概率论与数理统计.南京:南京大学出版社,1995.

王远(1980－),男,山东青岛人,博士生,主要研究领域为分布对象技术,构件技术,可信计算.

徐锋(1975－),男,副教授,博士,CCF 高级会员,主要研究领域为可信计算,电子商务安全.

吕建(1960－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为中间件技术,Agent 技术,分布式对象技术.

张林(1982－),男,硕士生,主要研究领域为信任管理,可信计算.