支持服务行为和质量的面向服务软件开发*)

殷 琴 李 俊 罗 毅 胡 昊 吕 建

(南京大学计算机软件新技术国家重点实验室,南京大学计算机软件研究所 南京 210093)

摘 要 面向服务的软件开发方法减少了开发应用程序的时间和复杂度。当前面向服务架构中的服务注册和发现主要关注于服务的接口描述和静态属性,却较少考虑服务行为和服务质量。对服务行为即外部可见的操作序列的忽视,可能导致服务产生运行时的异常和错误;忽视服务质量即其运行时的质量信息,则可能导致系统运行的低效。本文提出了一种扩展的面向服务的软件开发过程,探究服务交互过程中其行为一致和质量匹配。在面向服务的组件模型OSGi平台上,采用动态AOP技术,设计了一种支持服务行为和服务质量的中间件系统。该系统不但保证服务发现的调用一致性和服务替换的观察一致性,而且支持对服务运行时操作行为的检测和服务质量信息的收集,从而保证选择最佳的服务实现,提高应用系统的运行效率。

关键词 面向服务的软件开发,服务行为,服务质量

Enhancing Service-oriented Software Development Supporting Service Behavior and Quality

YIN Qin LI Jun LUO Yi HU Hao LU Jian

(State Key Laboratory for Novel Software Technology, Institute of Computer Software, Nanjing University, Nanjing 210093)

Abstract The service-oriented software development reduces the time and complexity of application development. However, in current service-oriented architecture, the service registration and discovery only focus on the interface and static properties of the service, but neglects the behavior and quality of the service. The ignorance of service behavior will result in some runtime errors and the ignorance of QoS will lead to the inefficiency of the application. This paper proposes an extension to service-oriented software development process, explores the consistency of service behavior and the match of QoS. In OSGi, a service-oriented component model, we adopt the technology of dynamic AOP and design a middleware to support the behavior and quality of service. In order to select the most suitable service implementation and to improve the efficiency of the application, the middleware assures the invocable and observable consistency, checks runtime service behavior and collects the QoS information.

Keywords Service-oriented software development, Behavior of service, Quality of service

1 引言

面向服务的软件开发近年来发展迅速并受到软件产业界和学术界越来越多的关注。面向服务的软件开发,以服务构件作为基本元素,服务构件的视图由服务规约定义,组装应用时只需要服务规约所定义的语法、语义和行为信息,而真正的服务提供者在运行时才集成到应用中[1]。服务规约的定义和运行时服务集成使得面向服务的开发支持运行时的服务发现、绑定和替换,而不需要重新启动甚至编译系统。

当前的服务规约主要关注服务的接口描述和属性信息,服务发现时通过精确的字段值或是语义的相似性匹配查找服务实现,这样可以保证服务的接口一致性。然而,由服务的行为即其外部可见的操作序列的不一致而引起的运行时刻的错误却不能避免;同时,由于缺少服务运行时的质量信息,例如可用性、响应时间、吞吐量等,使得查找返回的并不是当前最优的服务实现。

本文正是在这样的应用需求下,在OSGi(Open Service Gateway Initiative)[2]这一面向服务的平台上,采用动态AOP(Dynamic Aspect-Oriented Programming)[3]技术构建支持服务行为和服务质量的中间件系统。该架构在服务实现和服务使用之间增加服务代理层,该代理自动监测服务的运行,使其满足服务的行为约束;服务发现时,验证应用系统的行为需求和服务实现的行为约束之间的调用一致性;服务替换时,验证不同服务实现的行为约束之间的观察一致性;更进一步,该代理动态地记录和更新服务运行时的质量信息,支持基于动态的质量属性的服务发现和选择。

本文以下部分的内容作如下安排:第2部分分析当前的面向服务的软件开发过程及其存在的问题;第3部分通过添加服务的行为和质量属性重新定义服务规约,提出支持服务行为和质量的面向服务的软件开发过程,研究服务发现时的行为一致与质量匹配;第4部分在介绍OSGi平台和分析AOP技术的基础上,设计支持扩展服务的开发和集成的中间

*)本项目受国家863计划(2004AA112090,2005AA113160,2005AA113030)、国家973计划(2002CB312002)、国家自然科学基金(60273034,60233010,60403014)资助。殷 琴 硕士研究生,主要研究领域为软件工程、面向服务的体系结构;李 俊 硕士研究生,主要研究领域为普适计算;罗 毅 硕士研究生,主要研究领域为软件工程、软件体系结构;胡 昊 讲师,主要研究领域为软件过程、工作流技术、移动Agent技术;吕 建 教授,博士生导师,主要研究领域为对象技术、分布计算技术、移动agent技术。

件系统;相关工作和进一步的工作在第5部分和最后讨论。

2 面向服务的软件开发过程及其存在的问题

在典型的面向服务的交互中,包括3个组成部分:服务提供者,服务使用者和服务注册者。服务规约定义了服务的描述方式,通过接口和静态属性描述服务。面向服务的开发包括如下相关过程:

(1)发布:服务提供者实现服务,通过服务规约向系统注册者注册服务实现;

(2)部署:并行地,应用开发者即服务使用者依据服务接口描述,指定服务属性需求开发应用系统;

(3)查找:应用系统通过服务接口和属性需求向服务注册者查询服务,如果服务注册者找到满足需求的服务,则会返回这些服务提供者的描述列表;

(4)绑定:当服务使用者选择并与某一服务提供者绑定后,服务使用者得到服务实现对象的引用,具体的服务实现参与系统运行。

如上实现的面向服务的应用系统没有考虑服务行为和服务质量。

对服务行为的忽视可能导致服务产生运行时的错误。由于服务提供多个访问点即操作,对服务实现而言,服务行为即其操作序列需要满足一定的约束。相应地,服务使用者根据应用需要,调用服务操作的方式就是其对服务行为的需求。服务发现时,应用的行为需求与服务实现的行为约束的不匹配会导致调用不一致;新服务实现替换原有实现时,其行为约束之间的不匹配会引起观察不一致;在服务的运行过程中,服务操作的实际执行序列也可能不满足服务实现的行为约束。关于服务的行为不一致具体参见文[4,5]。

对服务质量信息的忽视可能导致系统运行的低效。由于基于服务的应用的增长以及众多的服务实现同样的接口,动态的服务质量信息成为区分服务实现和保证应用系统效率的重要指标。例如,在服务的运行过程中,被使用的服务的可用性决定了应用系统是否可以正常运行;针对某些网络应用系统,数据库服务的响应时间和吞吐量决定了该系统的运行效率。

3 支持服务行为和质量的面向服务的软件开发

面向服务的软件开发以服务构件为基本开发元素,应用系统通过服务规约进行组装,服务实现只在运行时才集成到系统中。服务是构件可见的访问点,构件在提供多个服务的同时也会使用多个外部提供的服务。服务的行为和质量信息使得能够进一步区分服务实现,选择最佳的服务实现,从而保证应用系统正确高效地运行。为了扩展面向服务的软件开发使之支持服务行为和服务质量属性,我们采用原有的服务构件描述方式,对服务规约进行扩展,使之能够描述服务行为和服务质量,并讨论服务的行为一致和质量匹配。

3.1 服务构件描述和服务规约

作为面向服务软件的基本元素,服务构件可以通过部署被其它应用使用,服务构件描述包括:

(1)构件名;

(2)提供的服务规约:定义了该构件提供给潜在用户的服务特征;

(3)需要的服务规约:定义了该构件需要由其它服务提供者提供的服务特征;

(4)组件属性:通过配置用以裁减和定制组件实现实例;

(5)实现文件:定义了组件的具体实现,如:Java类文件。

图1给出服务构件LogServiceImpl的实例描述。

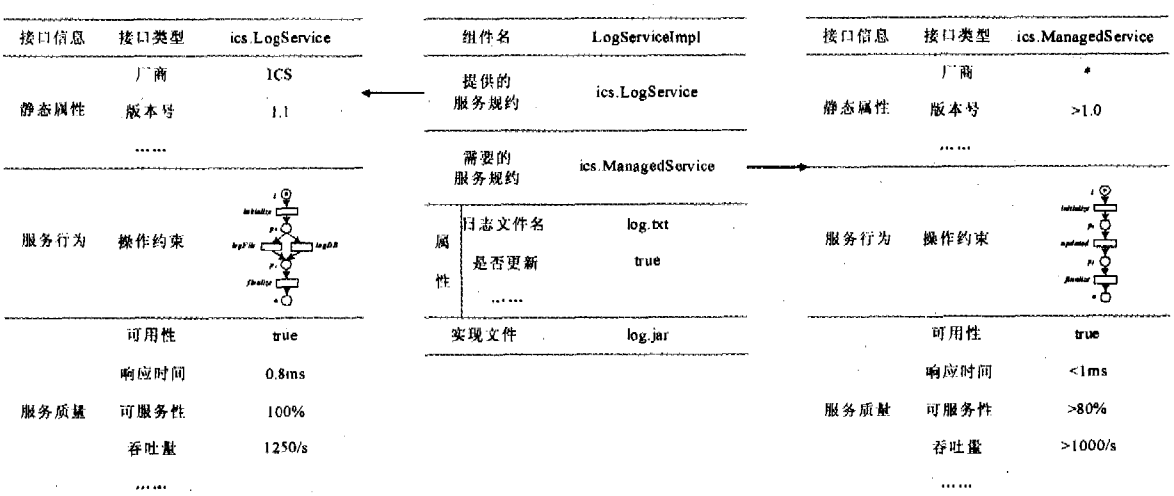

图1 服务构件实现实例

服务规约是服务构件的访问通道,通过添加服务的行为和质量属性,服务规约重新定义如下:

(1)接口信息。例如WSDL文件中的抽象定义或者为OSGi中的Java接口,这是区分服务的主要信息;

(2)静态属性。包括厂商、版本号信息等;

(3)行为信息。即服务外部可见的操作序列,可以通过标记的库所/变迁网描述,具体参见文[4]中的前期工作。

(4)质量属性。包括可用性、响应时间、可服务性(执行成功率)、吞吐量等等[6]。服务质量可用性表示当前服务是否存在于注册表中;响应时间度量了从发出请求到得到结果的时延,是处理时间和传输时间之和;可服务性记录了服务请求在预定时间内响应成功的比率;吞吐量描述了服务在某段时间内可响应的最大请求数。动态行为可以根据应用的需要加以扩充。

3.2 支持服务行为和质量的面向服务软件开发

为支持服务行为和质量属性,需要对面向服务的软件开发过程进行扩展,具体来说如图2所示。

(1)发布:服务提供者在注册服务实现时,向服务注册者

提供附加的行为约束和初始质量属性；

(2)部署：服务使用者开发应用系统时，指定服务行为和质量需求；

(3)查找：查找服务时，服务注册者除了匹配接口和静态属性值，还通过服务的行为一致性和质量匹配进一步过滤服务提供者列表；

(4)绑定：当与具体的服务实现绑定后，应用执行环境会实时监测服务的操作行为以保证其满足行为约束，同时动态更新质量属性，为以后服务发现和选择提供指导。

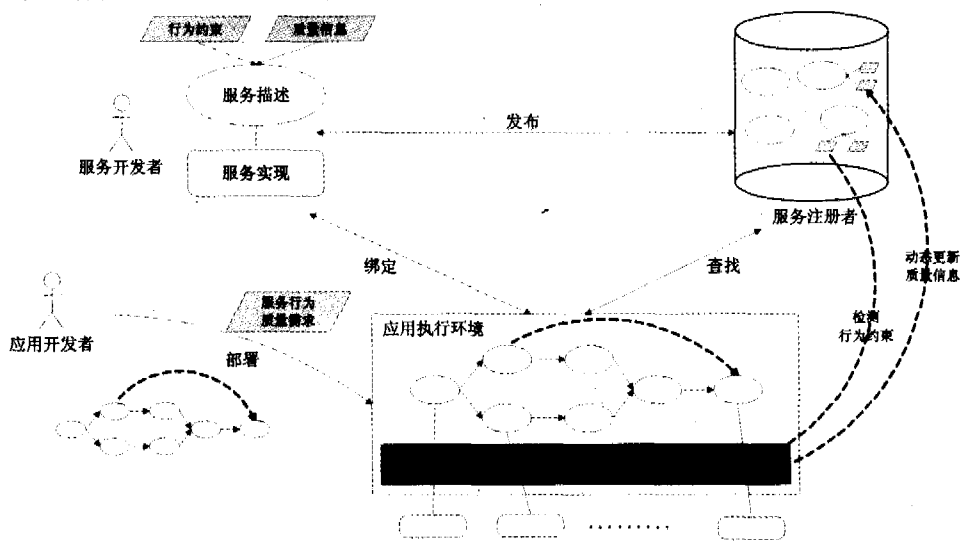

图 2　扩展的面向服务的软件开发过程

3.3　服务行为一致和服务质量匹配

在上述扩展的面向服务的软件开发过程中，服务查找需要进一步过滤服务提供者列表。返回的服务实现必须与服务使用者提出的需求相匹配，即不但匹配服务的接口和静态属性，而且匹配服务的行为和质量属性。具体说来，即接口信息一致，静态属性满足查询要求，行为满足一致性，质量属性也满足使用者的查询要求。我们着重讨论行为一致和质量匹配。

服务行为一致性的定义借鉴了 Ebert 和 Engles 为定义类及其子类行为描述的相互关系而提出的两种对象行为和相应的一致性关系：调用一致性和观察一致性[7]。在服务行为匹配中，所谓调用一致性是指把服务行为看成对服务操作的调用序列的描述，服务使用者在规约需求的行为图中所描述的操作调用序列，必须在返回的服务实现的行为图也可以被调用执行；所谓观察一致性是指把服务行为看成可观察操作序列的描述，若忽略新服务实现中增加的操作，新服务实现行为图中的可观察方法序列在原有服务实现的行为图中也可被观察。服务行为一致性验证视服务行为需求必须与服务实现提供的行为约束之间满足调用一致性，新服务实现与原有服务实现的行为约束之间必须满足观察一致性，具体定义见文[4]。

服务提供者在服务规约中给出服务质量属性的预定值，比如：服务可用、100%可服务、响应时间为处理时间、吞吐量为在局域网中可同时响应的最大请求数。事实上，这些质量属性在实际的运行环境中不断变化，因此需要有第三方代理收集服务的动态性能信息，并以此作为服务发现的重要指标。具体而言，服务使用者以查询语言的方式(例如 LDAP 语言)提供关于服务质量的需求，根据图 2 中的示例，服务质量查询条件如下：(availability＝true)&(responseTime＜1)&(servability＞0.8)&(throughput＞1000)；代理就需要在满足接口、静态属性、行为一致的服务实现中，查找满足使用者给定需求的服务，并按照服务质量属性优劣的顺序返回。

4　支持扩展服务的开发和集成的中间件系统

目前，开发支持服务行为和质量的应用系统非常复杂并且耗时，其原因是当前面向服务的架构并不支持服务行为一致和质量匹配，也没有相应的工具包帮助开发和部署服务的行为和质量属性。因此，我们使用动态 AOP 技术，在主流的面向服务的构件模型 OSGi 上，设计了支持服务行为和服务质量的中间件系统。该中间件系统包括一个构建在 OSGi 平台上的运行支持环境和一个以 Eclipse Plugin 形式提供的集成开发工具。

4.1　OSGi 服务平台

OSGi[2]是流行的面向服务的构件模型之一。开发 OSGi 上的应用遵循面向服务的开发方式，OSGi 联盟提出可以传输到本地网络和设备的开放的服务规约，服务提供者和应用开发者依照服务规约来进行软件开发。OSGi 平台以一致的方式为服务提供者、应用开发者、网关操作者以及设备提供商进行服务的开发、部署和管理提供了一个共有的架构。

OSGi 框架支持部署可扩展可下载的称为 bundle 的服务组件实现，一个 bundle 由 Java 中的类和其它资源文件构成，bundle 可以提供功能给最终用户或者提供服务给其它 bundle。服务由规约(这里规约指 Java 接口)定义，一个服务规约可以包含多个服务实现。服务对象是服务实现的主体，也是 bundle 之间的一种主要交互方式，由 bundle 注册并且通过多组名值对来描述服务实现。当服务发现时，动态注册者允许服务使用者通过提供完整的服务名和一个可选的基于 LDAP (Lightweight Directory Access Protocol)语法的过滤器对服务实现进行选择。OSGi 框架提供对服务依赖和 bundle 之间安全交互的管理。

4.2　动态 AOP

面向方面的程序设计(AOP)[8]是面向对象程序设计(OOP)的扩展，AOP 主要用来解决关注分离的程序模块化问题。从 AOP 的角度来看，应用程序除了包含功能逻辑还需要

关注许多横跨性需求，而实现这些横跨性需求的程序代码应该要从功能模块中分离，称为 Aspect。Aspect 和功能模块之间的连接通过插入点定义，并且通过织入机制将 Aspect 代码整合到功能模块中，从而满足系统的整体需求。使用 AOP 技术分离了不同关注的代码和系统功能代码，降低了软件系统的整体耦合度。

动态 AOP 是 AOP 的一种实现方式，它使用 Java 开发工具集的动态代理或者字节码处理技术在程序的运行过程中动态插入 Aspect。和 AOP 的原理一样，通过定义某些插入点，例如 before、after 或 around，在系统运行时动态地在方法的之前、之后和中间调用设定的 Aspect 程序。这样使得开发出来的程序具有动态的适应性和更低的耦合性。

4.3 中间件系统

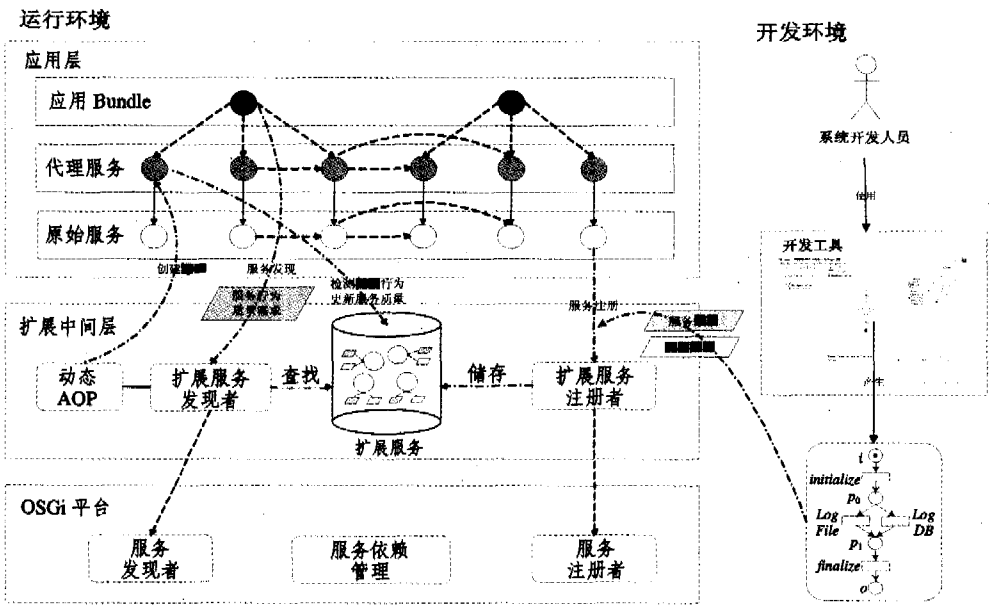

图 3 中间件系统结构

中间件系统采用动态 AOP 技术，支持在 OSGi 平台上开发保证服务行为一致和属性匹配的面向服务的应用软件。利用动态 AOP 技术的适应性和低耦合性的优点，在运行环境的服务层上封装扩展的中间层。软件开发人员通过使用基于 Eclipse 的开发工具开发出提供和需要扩展服务的 bundle，部署在 OSGi 平台上后，运行环境根据扩展服务规约保证应用系统中服务的行为一致和质量匹配。图 3 描述了中间件系统的结构。

4.3.1 运行环境

运行环境自底向上分为 OSGi 平台、扩展中间层和应用层 3 部分。OSGi 平台提供了一些用于部署可扩展和可加载的服务应用的设施。行为中间层扩展了原有的服务发现、注册和替换机制，加入了服务的行为和质量属性，并在运行时监测服务行为和更新服务质量。应用层包含了各种服务和应用的实现，它们通过行为中间层提供的功能去保障系统的行为安全。

OSGi 平台管理 bundle 的安装和更新，并且监控 bundle 和服务之间的依赖关系。平台的一个显著的功能是为 bundle 的开发者提供一个简洁一致的编程模型，通过减少服务规约和服务实现的耦合度来降低服务开发的复杂度。使用这样一个模型，开发人员在应用开发阶段只需要指定服务规约，把选择合适的服务实现推迟到系统运行时。

扩展中间层构建在 OSGi 平台之上，也遵循分离服务规约和实现的原则。扩展服务发现者和扩展服务注册者通过加入服务的行为和质量属性的方式封装了原有的交互机制。实现 bundle 在注册服务的时候加入服务的行为约束和预定质量属性，扩展服务注册者保存这些信息并进一步向 OSGi 平台注册服务。当服务发现时，应用可以提出服务行为和质量需求，扩展服务发现者根据底层平台返回的提供者列表，通过检验服务的行为一致和质量匹配找到最佳的服务实现，返回给应用。在服务首次被调用时，扩展服务发现者通过动态 AOP 创建这个服务实现的代理，代理服务通过定义 before 和 after 插入点，在服务使用的过程中，在这些插入点上检测运行时服务行为并且实时更新动态属性信息。

4.3.2 开发环境

由于 Eclipse[9] 是一个开放源代码的、基于 Java 的可扩展开发平台，通过逐步插入 Plugin 可以构造为功能强大的工具集，因此开发工具采用 Eclipse Plugin 的形式。开发工具提供典型的图形用户界面，开发人员可以通过拖放图形元素来定义服务行为，通过添加和赋值属性来描述服务质量属性。这种方式很大程度上降低了开发支持行为和质量的面向服务应用的复杂度。服务行为图和质量属性信息最终转化为文本方式，供扩展中间层在运行时使用。

在部署阶段，服务实现的行为和预定质量属性存储在 XML 格式的文件中。通过扩展 bundle 中 manifest 的头信息指明文件名，描述该服务实现的行为和质量属性的 XML 文件被包含在 bundle 中。描述文件中包括一组导出的服务及其服务行为和预定属性，一组导入的服务及其相应的行为和属性需求。为了与 OSGi 中标准的服务注册发现机制相兼容，继承了 OSGi 相关的标签，例如用于包含 LDAP 过滤器的 filter 标签等。

5 相关工作和总结

大部分的面向服务的软件开发都是在一个基于 Web 的

面向服务计算的基础设施上进行，这个基础设施由描述服务接口的 WSDL，定义服务交互的 SOAP 和支持服务注册发现的 UDDI 标准组成。服务提供者在公共的 UDDI 服务注册系统中注册服务，服务使用者向注册系统查询发现服务主要都集中于服务的接口和静态属性在语法和语义上的匹配[10]。针对服务实现数量不断递增，许多服务实现提供重复的功能这一情况，很多研究工作都致力与进一步区分服务实现，发现满足约束要求的最佳服务。

文[11]提出的中间件平台在满足使用者需求和组合服务的结构的前提下，考虑服务质量以达到使用者满意度的最大化，值得借鉴的是描述和比较了局部最优和全局最优的两种服务选择方法。文[6]则同时考虑服务和网络性能，并给出较为完整的服务质量属性的定义。这些工作都关注服务质量属性，但是上述服务没有考虑服务操作的行为约束。我们的中间件系统则支持对服务行为的描述，通过分析服务操作的调用序列，验证服务调用一致性和观察一致性关系，监测服务执行过程中对操作时序约束的满足。

基于面向服务的组件模型 OSGi，Cervantes 提出服务组件模型的概念，并实现 Gravity 来管理未来上下文感知应用中的服务动态可用性和服务依赖[12]。Frei 设计了一个用在 Ad hoc 体系下的动态轻量级的平台，使用动态 AOP 的代理技术，向用户屏蔽服务依赖管理和服务无缝切换[3]。然而，这些在 OSGi 平台上的一些服务管理的系统并没有关注服务的静态和动态的行为。

本文的工作乃是我们前期工作的深化和延续。本文的贡献首先在于扩展了服务规约使之支持服务的行为和质量属性，提供开发环境支持扩展属性的定义，扩展 OSGi 平台提供支持服务行为一致、质量匹配的运行环境。本文的贡献还在于把动态 AOP 的思想引入面向服务开发的领域，为在扩展中间件层支持服务行为提供了一个独立透明的解决方案。

进一步工作 未来的工作主要集中在中间件系统的改进上。当前的中间件系统不支持多个服务的组合和协同，因此，未来的中间件系统将基于服务的行为和质量，设计出一种行为一致的、质量全局最优的服务组合和协同的模式，并在运行环境和开发工具上给予支持。此外，开发环境基于 Eclipse Plugin 设计，但与 Eclipse 中的 JDT 分离，这使得开发人员在实现服务接口和定义服务的行为与质量的时候需要使用两种不同的工具。因此，结合 JDT 和服务行为及属性开发工具可以简化面向扩展服务的应用的开发。

另一项工作包括将 Ketfi[13] 提出的服务接口的动态适应方法应用到服务发现、替换和组合中。动态适应性通过在服务接口中使用映射和组合机制，避免了不同软件实体之间的语法依赖。进一步，在动态适应过程中引入服务行为也很有意义，我们可以对服务动态适应的映射和组合进行行为描述，并在中间件系统中提供支持。

参 考 文 献

1 Papazoglou M P, Georgakopoulos D. Service-oriented Computing: Introduction. Communications of the ACM, 2003, 46(10): 24～28

2 The Open Services Gateway Initiative (OSGi), www.osgi.org

3 Frei A, Alonso G. A dynamic lightweight Platform for Ad-hoc Infrastructures. In: Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), Kauai Island, Hawaii, March 2005

4 Yin Qin, Hu Hao, Li Jun, et al. A behavior consistent service discovery and substitution mechanism in osgi. In: Proceedings of the Ninth IASTED International Conference on Software Engineering and Applications, 2005

5 Yin Qin, Hu Hao, Li Jun, et al. An Approach to Ensure Service Behavior Consistency in OSGi. In: Proceedings of the Asia-Pacific Software Engineering Conference (APSEC), 2005

6 Tian M, Gramm A, Naumowicz T, et al. A Concept for QoS Integration in Web Services. In: 1st Web Services Quality Workshop (WQW2003)

7 Ebert J, Engels G. Observable or Invocable Behavior - You Have to Choose: [Technical Report]. 94-38. Department of Computer Science, Leiden University, December 1994

8 Kiczales G. Aspect-Oriented Programming. In: Proceedings of the European Conference on Object-Oriented Programming (ECOOP). Springer-Verlag, Finland 1997

9 www.eclipse.org

10 Alonso G, Casati F, Kuno H, et al. Web Services: Concepts, Architectures and Applications. Springer-Verlag, Berlin Heidelberg New York, 2004

11 Zeng Liangzhao, Benatallah B, Ngu A H H. QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Software Engineering, 2004, 30(5): 311～327

12 Cervantes H, Hall R S. Autonomous Adaptation to Dynamic Availability Through A Service-Oriented Component Model. In: International Conference on Software Engineering (ICSE), Edinburgh, Scotland, May 2004

13 Ketfi A, Belkhatir N. Dynamic Interface Adaptability in Service Oriented Software. In: Eighth International Workshop on Component-oriented Programming (WCOP'03), 2003