WEBTRANSACT: UMA INFRAESTRUTURA PARA ESPECIFICAÇÃO E COORDENAÇÃO DE COMPOSIÇÕES DE SERVIÇOS WEB ROBUSTAS

Paulo de Figueiredo Pires

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profª. Marta Lima de Queirós Mattoso, D.Sc.

Prof. Mário Roberto Folhadela Benevides, Ph.D.

Prof. Marco Antônio Casanova, Ph.D.

Profª. Louiqa Raschid, Ph.D.

Prof. Asterio Kiyoshi Tanaka, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
JUNHO DE 2002

PIRES, PAULO DE FIGUEIREDO

WEBTRANSACT: Uma Infraestrutura para Especificação e Coordenação de Composições de Serviços Web Robustas [Rio de Janeiro] 2002

VIII, 42 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2002)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Composição de Serviços Web

2. WSDL

3. Composição de Serviços Eletrônicos

4. Transações na Web

I. COPPE/UFRJ II. Título (série)

À Flávia, com amor.

AGRADECIMENTOS

À professora Marta Mattoso, pela orientação, apoio e incentivo que recebi durante todo o desenvolvimento deste trabalho e ainda, por confiar no meu trabalho e ter me aberto as portas para novas oportunidades profissionais.

Ao professor Mário Benevides, pela orientação, incentivo e por ter me proporcionado a felicidade de conhecer uma nova área de conhecimento.

Ao professor Cláudio Esperança, pelo auxílio essencial que me prestou para viabilizar o meu doutorado sanduíche.

À professora Louiqa Raschid, por suas sugestões e comentários valiosos durante o período do doutorado sanduíche no qual trabalhamos juntos na Universidade de Maryland e ainda, por ter vindo ao Brasil para participar como membro da banca deste trabalho.

Aos membros da banca, professores Casanova, Tanaka e Valmir, pela revisão do trabalho, críticas construtivas, e incentivos.

À todos os colegas de doutorado, em especial à Fernanda, Vivacqua e Bevi, pela amizade e companheirismo.

Aos professores Jano e Blaschek, pelas oportunidades que me foram dadas.

À Cláudia Prata, Patrícia, Solange e Mercedes por terem me ajudado sempre que precisei.

Ao CNPq, Capes, Faperj e Fundação COPPETEC pelo fundamental apoio financeiro.

E em especial à Flávia que me acompanhou durante todos os longos dias dedicados à confecção deste trabalho, me dando amor, carinho, apoio e companheirismo (até mesmo revisando os meus textos!) quando eu mais precisei. Normalmente, o trabalho de desenvolvimento de uma tese é um trabalho solitário. Agora, eu tenho a felicidade de dizer que esse trabalho não o foi. Flávia, graças a você.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

WEBTRANSACT: UMA INFRAESTRUTURA PARA ESPECIFICAÇÃO E COORDENAÇÃO DE COMPOSIÇÕES DE SERVIÇOS WEB ROBUSTAS

Paulo de Figueiredo Pires

Junho/2002

Orientadores: Marta Lima de Queirós Mattoso
Mário Roberto Folhadela Benevides

Programa: Engenharia de Sistemas e Computação

A recente evolução das tecnologias para a internet, principalmente devido ao surgimento da *Extensible o Markup Language* (XML) e suas tecnologias relacionadas, está estendendo o papel da *World Wide Web* de interação de informação para interação de serviço. Esta próxima onda da era da internet está sendo conduzida por um conceito denominado *serviços Web (Web services)*. A tecnologia de serviços Web provê o suporte necessário para uma nova oportunidade de negócio, i.e., a possibilidade de prover serviços agregados através da composição de serviços Web básicos. Neste trabalho, nós apresentamos uma infraestrutura, denominada *WebTransact*, para a construção de composições de serviços Web seguras, manuteníveis, e escaláveis. A infraestrutura WebTransact é composta por uma arquitetura em múltiplas camadas, uma linguagem baseada em XML, e um modelo de transação. A arquitetura em múltiplas camadas da WebTransact separa a tarefa de agregar e homogeneizar serviços Web heterogêneos, da tarefa de especificar padrões de interação de transação, provendo um novo e genérico mecanismo para tratar a complexidade introduzida pela existência de um grande número de serviços Web. A linguagem baseada em XML, *Web Services Transaction Language* (WSTL), é usada para descrever o suporte a transações e o conteúdo de serviços Web, para definir as regras de agregação de serviços Web, e para especificar os padrões de interação de transação de composições. WSTL é uma extensão do WSDL logo, ela é aderente aos padrões XML que constituem a tecnologia de serviços Web. O modelo de transação provê as garantias de correção adequadas, para a execução de composições de serviços Web construídas com a WSTL.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

WEBTRANSACT: A FRAMEWORK FOR SPECIFYING AND COORDINATING RELIABLE WEB SERVICES COMPOSITIONS

Paulo de Figueiredo Pires

June/2002

Advisors: Marta Lima de Queirós Mattoso
Mário Roberto Folhadela Benevides

Department: Systems Engineering and Computer Science

The recent evolution of internet technologies, mainly guided by the Extensible Markup Language (XML) and its related technologies, are extending the role of the World Wide Web from information interaction to service interaction. This next wave of the internet era is being driven by a concept named *Web services*. The Web services technology provides the underpinning to a new business opportunity, i.e., the possibility of providing value-added services through the composition of basic Web services. In this work, we present a framework, named *WebTransact*, for building reliable, maintainable, and scalable Web service compositions. The WebTransact framework is composed of a multilayered architecture, an XML-based language, and a transaction model. The multilayered architecture of WebTransact separates the task of aggregating and homogenizing heterogeneous Web services from the task of specifying transaction interaction patterns, thus providing a new general mechanism to deal with the complexity introduced by a large number of Web services. The XML-based language, named *Web Service Transaction Language* (WSTL), is used for describing the transaction support and the content of Web services, for defining the aggregation rules of Web services, and for specifying the transaction interaction patterns of compositions. WSTL is an extension of WSDL thus it is adherent to the XML-based standards that enable Web service technology. The transaction model provides the adequate correctness guarantees when executing Web services compositions built with WSTL.

Índice

FIGURAS

1. Introdução

Neste Capítulo, nós apresentamos a motivação, os objetivos, e o escopo desta tese. Na Seção 1.1, apresentamos a evolução das tecnologias Web que vem viabilizando o surgimento de uma nova categoria de processos de negócio, denominada *composição de serviços Web* (*Web service composition*). Na Seção 1.2, levantamos os novos requisitos necessários para o desenvolvimento de composição de serviços Web. Na Seção 1.3, definimos qual é o objetivo, bem como, o escopo desta tese e apresentamos como nós endereçamos cada um dos requisitos levantados na Seção 1.2. Finalmente, na Seção 1.4 delineamos a organização deste trabalho.

1.1 MOTIVAÇÃO

Desde o seu surgimento, a Web vem sendo utilizada, principalmente, para acessar documentos e aplicações de forma interativa. Na maior parte das vezes, esses acessos tem sido feitos por usuários humanos, normalmente utilizando *Web browsers*. Contudo, com a evolução recente das tecnologias para internet, principalmente da linguagem XML (*Extensible Markup Language*) [139] e suas tecnologias relacionadas, o papel da Web vem evoluindo de um ambiente baseado na troca de informação para um ambiente baseado na troca de *serviços*. Esta nova onda da era da internet esta sendo guiada pelo conceito de *serviços Web* (*Web Services*).

Serviços Web podem ser definidos como programas modulares, independentes e auto-descritivos, que podem ser descobertos e invocados através da internet ou de uma intranet coorporativa. Através da tecnologia de Serviços Web, pode-se encapsular processos de negócios pré-existentes, publicá-los como serviços, descobrir dinamicamente serviços publicados, e trocar informações que ultrapassem as fronteiras de uma corporação. Serviços Web são a tecnologia chave para promover a interoperabilidade entre aplicações através da *World Wide Web*.

Serviços Web combinam aspectos de desenvolvimento baseado em componentes distribuídos ([62], [83], [110]) e a *World Wide Web*. Da mesma forma que componentes distribuídos, serviços Web expõem uma interface que pode ser reutilizada sem que seja

necessário se preocupar com a forma como o serviço é implementado. Diferentemente das tecnologias atuais de desenvolvimento de componentes distribuídos, serviços Web não são acessados por protocolos dependentes de um modelo de objetos específico, como o *Distributed Component Object Model* (DCOM) [84], o *Remote Method Invocation* (RMI) [127], ou o *Internet Inter-ORB Protocol* (IIOP) [97]. Ao invés de utilizar protocolos ligados a determinada tecnologia, serviços Web são acessados por protocolos e formato de dados ubíquos, como o *Hypertext Transfer Protocol* (HTTP) e o XML, ambos padrões independentes de fabricante e largamente utilizados na Web. Esta é uma diferença fundamental entre serviços Web e as tecnologias atuais que implementam o desenvolvimento baseado em componentes distribuídos. Devido ao uso de protocolos e formato de dados ubíquos, os serviços Web atingem um grau mais elevado de interoperabilidade com relação às tecnologias atuais. A tecnologia de serviços Web atual pode ser considerada como uma arquitetura cliente-servidor aberta que possibilita a interoperabilidade entre sistemas diferentes sem utilizar qualquer biblioteca ou programa proprietário no cliente. Desta forma, o desenvolvimento de aplicações cliente-servidor tradicionais é simplificando através da efetiva eliminação de dependências tecnológicas entre programas cliente e servidor.

Na realidade, serviços de Web são uma evolução dos padrões e protocolos existentes para a Web. O padrão HTML [140] que permite a exibição do conteúdo de páginas Web, evoluiu para um padrão mais generalizado, o XML [139] que pode ser usado não só para descrever o conteúdo de páginas Web como também para representar praticamente qualquer tipo de estrutura de dados. O protocolo HTTP ([36], [95]) foi estendido dando origem a um novo protocolo denominado *SOAP*[1] [145], o qual provê um modelo de comunicação mais generalizado que possibilita a troca de mensagens entre dois sistemas independentes. *Web Services Description Language* (WSDL) [138] é uma linguagem XML para a descrição da interface de um serviço Web que permite a um determinado programa interagir com um serviço Web. Serviços Web podem ser publicados em um diretório público e, a partir desta publicação, serem descobertos e utilizados por outros programas através de uma nova especificação denominada *Universal Description, Discovery, and Integration* (UDDI) [134], a qual evoluiu da especificação *Lightweight Directory Access Protocol* (LDAP) [150].

[1] Até a versão 1.1 SOAP era o acrônimo de *Simple Object Access Protocol*. Contudo, desde a versão 1.2, SOAP não é mais um acrônimo.

Serviços Web são tipicamente construídos utilizando-se as especificações XML, SOAP, WSDL, e UDDI. Atualmente, a maioria das grandes organizações de desenvolvimento de software está implementando ferramentas baseadas nestes novos padrões para Web. Como exemplo podemos citar: a plataforma *.Net* da Microsoft ([88], [89]), a plataforma da IBM, *Dynamic E-business* ([57], [58]); a plataforma da Sun, *Open Net Environment* (Sun ONE) ([124], [126]), a plataforma *WebLogic Integration 2.0* da BEA systems [9], e a plataforma *Orbix E2A Web services integration* da IONA [60]. Considerando a rápida velocidade na qual implementações destes padrões estão sendo disponibilizadas, juntamente com o claro compromisso de várias e importantes empresas de software, nós acreditamos que, brevemente, tais padrões serão tão amplamente implementados como o padrão HTML é implementado atualmente.

De acordo com o cenário que acabamos de descrever, um número crescente de serviços on-line serão publicados na Web durante os próximos anos. Conforme estes serviços se tornarem disponíveis no ambiente de serviços Web, uma nova oportunidade de negócios será criada, i.e., a possibilidade de fornecimento de serviços Web complexos, construídos a partir da agregação de serviços Web disponíveis nas Web [21].

Composição de serviços Web pode ser definida com sendo a habilidade de uma determinada empresa fornecer serviços a seus clientes através da composição de serviços Web básicos, possivelmente desenvolvidos por diferentes corporações ([22], [54], [21]). A composição de serviços Web compartilha muitos dos requisitos existentes na área de gerência de processos de negócios (*business process management*) ([4], [64], [117], [48]). Em ambos os sistemas existe a necessidade de: coordenar a seqüência correta de chamadas de determinado serviço dentro de uma composição; administrar os fluxo de dados entre serviços; e administrar a execução de composições como unidades de transação. Além disto, ambos os sistemas precisam garantir alta disponibilidade, robustez, e escalabilidade. Contudo, a tarefa de construir composições de serviços Web é muito mais complexa devido ao grau de *autonomia*, *heterogeneidade*, e *dinamismo* existente no ambiente de serviços Web. Diferentemente dos componentes de processos de negócio tradicionais, os serviços Web são tipicamente desenvolvidos e gerenciados por diferentes organizações, além de serem projetados independentemente de qualquer entidade coletiva de computação. Considerando que cada organização tem suas próprias regras empresariais, serviços Web devem ser tratados como unidades estritamente autônomas. A heterogeneidade de serviços Web se manifesta por diferenças estruturais e semânticas que

podem existir entre serviços Web semanticamente equivalentes. Em um ambiente de serviços Web é bastante provável que diferentes serviços Web ofereçam a mesma funcionalidade semântica assim, a tarefa de construir composições destes serviços deve, de alguma maneira, considerar este problema. A construção de composições de serviços Web é ainda complicada pelo fato de que o ambiente de serviços Web é altamente dinâmico. A Web é um ambiente verdadeiramente dinâmico e os Serviços Web não serão uma exceção. Em um curto espaço de tempo, novos serviços Web serão publicados, por outro lado, serviços Web publicados podem ficar indisponíveis no mesmo espaço de tempo. Além disso, serviços Web não deverão expor um comportamento estático, simplesmente porque eles terão que evoluir de acordo com as regras de negócio do mercado associado ao serviço.

A composição de serviços Web fornece a fundação necessária para o desenvolvimento orientado a serviços na Web. Contudo, a construção de serviços agregados neste ambiente não é uma tarefa trivial. Devido às muitas singularidades do ambiente de serviços Web, não é possível se basear nos modelos e soluções atualmente utilizados para construir e coordenar composições de serviços Web.

1.2 REQUISITOS PARA O DESENVOLVIMENTO DE COMPOSIÇÕES DE SERVIÇOS WEB

A WSDL viabiliza a interoperabilidade entre processos que são acessados via Web, fornecendo a infraestrutura básica para construção de composição de serviços Web. Contudo, somente a viabilização da comunicação ponto-a-ponto entre programas clientes e serviços Web não é suficiente para o desenvolvimento de composições de serviços Web que possuam as características necessárias de robustez, manutenibilidade, e escalabilidade. A construção de composições robustas necessita de mecanismos para a coordenação de transações que envolvam múltiplos e *autônomos* serviços Web. Composições manuteníveis e escaláveis necessitam de um mecanismo de acoplamento fraco para conectar serviços Web a composições, o qual deve ser capaz de isolar os problemas derivados da *heterogeneidade* e *dinamismo* inerente às interfaces dos serviços Web. Ainda, a tarefa de desenvolver composições pode ser sensivelmente otimizada se o desenvolvedor de composições não for obrigado a resolver, diretamente, os problemas relativos a heterogeneidade e dinamismo das interfaces de serviços Web. Desta forma, uma infraestrutura para desenvolvimento de composições de serviços Web deve fornecer

mecanismos para homogeneizar e isolar tais interfaces antes que elas sejam utilizadas na implementação de composições. Até o presente momento, a plataforma para desenvolvimento de serviços Web não possui uma infraestrutura que forneça soluções para esses problemas.

Nas próximas Seções, nós apresentamos a nossa visão dos requisitos necessários para suporte de transações e homogeneização de serviços no contexto de composições de serviços Web.

1.2.1 Coordenação de Transações no Contexto Web

Tradicionalmente, uma transação deve satisfazer as propriedades de *atomicidade*, *isolamento*, *consistência*, e *durabilidade* conhecidas com propriedades ACID ([46], [14], [77]). Estas propriedades são utilizadas para definir o conceito de correção dos protocolos e modelos de transações tradicionais. No ambiente de serviços Web, fornecer mecanismos que suportem todas as propriedades ACID não é desejável ou mesmo possível.

O conceito tradicional de *atomicidade* significa que todos os passos executados durante uma transação devem ser executados como uma unidade, ou seja, ou todos os passos são executados com sucesso, ou nenhum deles é executado. Esta noção rígida de atomicidade não é razoável em um ambiente de serviços Web. Uma composição deve explorar os diferentes tipos de suporte à transação fornecidos pelos serviços Web bem como o fato de que, no nível semântico, um mesmo serviço pode ser implementado por múltiplos e, sintaticamente, diferentes serviços Web. Por exemplo, se um determinado serviço Web suporta compensação, então uma composição, na qual este serviço participe, deve explorar esta característica para contornar problemas decorrentes de falhas que possam vir a ocorrer durante a sua execução. O mesmo deve ser feito quando um determinado serviço semântico é suportado por mais de um serviço Web. Sempre que um determinado serviço Web falhar, deve ser verificado se não existe um outro serviço Web que suporte a semântica do serviço em questão. Desta forma, uma composição de serviços Web pode atingir os seus objetivos mesmo quando alguns de seus passos (Serviços Web) não são executados com sucesso.

A propriedade de *consistência* significa que uma transação deve resultar em mudanças de estado consistentes, ou seja, uma determinada transação executada sobre um

sistema cujo estado seja consistente levará este sistema para outro estado, também consistente. De forma a garantir a consistência através de múltiplos serviços Web, os subsistemas locais devem ser monitorados para identificar chamadas a serviços que violem as dependências globais [115]. Tal monitoramento não é viável no ambiente de serviços Web devido à rígida autonomia e também devido ao potencial grande número de Serviços Web.

A propriedade de *isolamento* define que o controle de execução concorrente de transações deve fornecer a ilusão de que transações concorrentes são executadas segundo uma ordem serial. Geralmente, mecanismos de bloqueio (*locking*) são utilizados para suportar a propriedade de isolamento. Como serviços Web podem possuir suporte a transações dissimilares, o sistema local que implementa o serviço Web pode não suportar ou permitir bloqueios quando o serviço Web é acessado. Além disto, garantir o isolamento pode ter um alto custo porque a execução transacional de uma composição pode ter longa duração, e fornecer isolamento para transações de longa duração levaria a deteriorização da eficiência dos subsistemas que implementam os serviços Web.

A única propriedade que deve ser totalmente suportada em composições transacionais é a durabilidade. O conceito tradicional de durabilidade significa que quando uma transação completa a sua execução, todas as suas ações tornam-se persistentes, mesmo na presença de falhas. Da mesma forma, a durabilidade deve ser assegurada no ambiente de serviços Web.

De acordo com a discussão levantada nos parágrafos anteriores, um modelo de transações para coordenação da execução de composições de serviços Web deve garantir a propriedade de durabilidade, bem como permitir uma forma relaxada de atomicidade que considere o suporte dissimilar de transação presente nos serviços Web e também a existência de serviços Web semanticamente equivalentes.

Existem vários modelos de transações que propõem o relaxamento (de alguma) das propriedades ACID ([30], [111]). Estes modelos de transação, conhecidos como *Modelos de Transações Estendidas* (*Extended Transaction Models - ETM*), tem sido propostos para suportar aplicações específicas que requerem modelos de transação mais flexíveis que aqueles baseados nas propriedades ACID. Como exemplos de ETMs podemos citar: modelo de transações aninhadas fechado (*closed nested transaction*) [93], sagas [42],

contracts [146], transações flexíveis (*flexible transaction*) [32], e multitransaction [16]. Contudo, os ETMs não endereçam todos os requisitos transacionais necessários à coordenação da execução de composições de serviços Web.

Três são os problemas relacionados à adoção de ETMs para coordenar a execução de transações em composições de serviços Web. Primeiro, ETMs não foram desenvolvidos para coordenar entidades de software semanticamente equivalentes, e sim exibindo interfaces e comportamento dissimilares, o que deve ocorrer no ambiente de serviços Web. Segundo, ETMs têm sido desenvolvidos levando-se em consideração que todas as entidades de software envolvidas fornecem suporte para um determinado conjunto de funcionalidades transacionais. Por exemplo, ETMs para sistemas de gerência de banco de dados heterogêneos (SGBDH) [109] requerem que cada subsistema participante exponha uma interface que suporte o protocolo *two-phase commit* (2PC) ([45], [70], [69]), restringindo desta forma as entidades de software participantes a sistemas de banco de dados. No ambiente de serviços Web esta funcionalidade pode não estar presente, já que estamos lidando com serviços que rodam dentro de sistemas arbitrários que podem não possuir ou expor interfaces transacionais. Terceiro, todos os ETMs têm sido desenvolvidos para suportar de alguma forma a propriedade de consistência, o que não é viável em uma ambiente como de serviços Web. Devido aos fatores que acabamos de descrever, os ETMs não podem ser utilizados diretamente para coordenar execuções transacionais de composições de serviços Web. Logo, existe a necessidade de desenvolver-se novos modelos de transação, específicos para este tipo de ambiente.

Para concluir esta Seção, nós sumarizamos os requisitos para coordenação de transações em ambientes de serviços Web:

- **Suporte a transações dos provedores de serviços Web.** Provedores de serviços Web são independentes e autônomos, não é desejável ou mesmo possível depender do suporte a transação oferecido por provedores de serviços Web. O modelo de transações não pode esperar que os provedores de serviços Web ofereçam funcionalidades transacionais como uma interface 2PC pública.

- **Serviços Web ofereceram suporte a transação dissimilar**. O modelo de transações deve explorar o suporte a transações específico oferecido por cada provedor de serviço Web.

- **Serviços Web semanticamente equivalentes**. Como o mesmo serviço semântico pode ser suportado por diversos e diferentes serviços Web, o modelo de transações deve explorar esta característica para aumentar a robustez das composições.

1.2.2 Homogeneização de Serviços Web

Um outro aspecto importante que deve levado em consideração em uma ambiente de serviços Web é a agregação de serviços Web semanticamente equivalentes. Com o crescimento do número de serviços Web disponíveis, é provável que apareçam vários serviços Web diferentes, implementados por diferentes empresas, fornecendo a mesma funcionalidade semântica. O processo de agregação de serviços Web envolve a resolução de *dissimilaridades semânticas* que podem existir entre estes serviços e a descrição da *capacidade de resposta* do serviço Web. Por exemplo, considere dois serviços Web fornecendo a funcionalidade de reserva de carros em duas empresas diferentes. É pouco provável que a comunicação com ambos os serviços se dê utilizando o mesmo formato de mensagem. As dissimilaridades que podem ocorrer na agregação de serviços Web podem variar de diferenças de nome a diferenças estruturais. O problema relativo a solução de dissimilaridades semânticas de serviços é similar ao problema relativo a solução de dissimilaridades de dados existentes em sistemas baseados em *Mediadores* ([18], [76], [41], [106], [133]) e SGDBHs ([18], [31], [109]). As técnicas empregadas na solução de dissimilaridades de dados podem ser adaptadas para resolver dissimilaridades semânticas. Outros trabalhos já abordam, de alguma forma, o problema de resolução de dissimilaridades semânticas na Web através do emprego da tecnologia de tradutores (*Wrapper mediator*) ([49], [136]). É importante ressaltar que tais técnicas não podem ser *diretamente* empregadas na resolução de dissimilaridades semânticas de serviços. O problema de resolução de dissimilaridades de dados é normalmente relacionado à homogeneização de diferentes esquemas de dados enquanto que o problema de resolução de dissimilaridades de serviços está relacionado à homogeneização de diferentes interfaces de acesso a serviços (formato de mensagens).

Além da dissimilaridade semântica, serviços Web são diferentes com relação a sua capacidade de resposta (conteúdo). Por exemplo, um serviço de reservas de carros de uma empresa *A* pode ser capaz de efetuar reservas somente no Brasil, enquanto que outro serviço de reservas de carros de uma empresa *B* pode ser capaz de efetuar reservas mundialmente. Tal informação é particularmente importante durante a execução de composições de serviços Web. Se a capacidade de resposta dos serviços Web está disponível, é possível selecionar somente aqueles serviços Web que são capazes de responder a determinada execução, evitando chamadas desnecessárias a outros serviços Web.

Se considerarmos o número de organizações atualmente conectadas à internet, podemos notar que existe um forte potencial latente para o surgimento de um grande número de serviços Web em um futuro próximo. Desta forma, para que composições possam ser construídas de modo adequado, é imperativo que sejam desenvolvidos mecanismos que facilitem a agregação destes serviços Web. A agregação de serviços Web visa facilitar o desenvolvimento de suas composições. Um determinado processo de negócio pode ser construído a partir de serviços disponibilizados em uma camada de serviços Web agregados. Esta camada, por sua vez, tem como objetivo prover isolamento de dissimilaridades, e até mesmo da existência de diversos e diferentes serviços Web (semanticamente equivalentes), fornecendo uma interface única e padrão, a qual pode ser utilizada para a construção de composições. Desta forma, o desenvolvedor de composições de serviços Web pode concentrar-se no desenvolvimento das regras de negócios relacionados à composição, sem ter que se preocupar com o comportamento específico de cada serviço Web disponível. Em um ambiente de serviços Web, caracterizado pela grande quantidade de serviços disponíveis, a falta de mecanismos para organizar e homogeneizar estes serviços aumenta, de forma significativa, a complexidade de desenvolvimento de composições baseadas em tais serviços.

1.3 OBJETIVO E ESCOPO DESTE TRABALHO

O principal objetivo deste trabalho é o desenvolvimento e especificação de uma infraestrutura para a construção de composições de serviços Web robustas, manuteníveis,

e escaláveis. De forma a atingir este objetivo, nós desenvolvemos a *WebTransact*[2] [108]. A WebTransact é uma infraestrutura que disponibiliza mecanismos de suporte a construção de composições de serviços Web, as quais tratam dos problemas específicos ao ambiente de serviços Web. Esta infraestrutura é composta por uma arquitetura em múltiplas camadas, uma linguagem baseada em XML, e um modelo de transações.

A *arquitetura em múltiplas camadas* [108] é composta por componentes especializados que tratam do problema de homogeneização de serviços Web. A arquitetura WebTransact é baseada na tecnologia de mediadores. Sistemas baseados em mediadores ([18], [76], [41], [106], [133]) têm sido desenvolvidos com sucesso com a finalidade de mediar a capacidade de consulta de repositórios remotos (gerenciados ou não por bancos de dados) e fornecer uma visão integrada da informação, no nível do mediador. Na WebTransact, estas técnicas foram adaptadas para resolver o problema de *dissimilaridade semântica* e para descrever a *capacidade de conteúdo* dos serviços Web.

A *linguagem baseada em XML* [108], denominada *Web Service Transaction language* (WSTL), foi desenvolvida para: descrever o suporte de transações e conteúdo dos serviços Web; definir as regras de agregação de serviços Web semanticamente equivalentes; e especificar os padrões de interação de serviços Web em composições. A WSTL, desenvolvida no contexto desta tese, é uma extensão do WSDL, desta forma, ela é aderente aos padrões XML que constituem a tecnologia de serviços Web.

O *modelo de transações* [108] proposto pela WebTransact fornece os critérios de correção adequados à execução de composições de serviços Web construídas em WSTL.

Na WebTransact, composições são construídas sobre uma camada de serviços Web homogeneizados. Desta forma, o desenvolvedor de composições não tem que se preocupar com as dissimilaridades quanto ao suporte a transações bem como àquelas relacionadas à semântica ou ao conteúdo de diferentes serviços Web. Tais dissimilaridades são solucionadas quando os serviços Web são integrados à arquitetura WebTransact.

[2] As idéias centrais da infraestrutura WebTransact tiveram origem na arquitetura MedTransact [107]. A MedTransact foi desenvolvida durante o ano 2000 quando eu estava tarbalhando com a professora Louiqa Raschid como pesquisador visitante no laboratório Clip da *University of Maryland*.

Os desenvolvedores de composições utilizam a linguagem WSTL para especificar composições de serviços Web. Os construtores fornecidos pela WSTL permitem não apenas a especificação do fluxo de controle e de dados entre serviços Web, mas também a especificação de diferentes níveis de robustez e otimização para uma determinada composição transacional de serviços Web. Por exemplo, um desenvolvedor pode definir de forma explícita, através destes construtores, que ação deve ser tomada no caso de uma chamada a um determinado serviço Web falhar. Além disso, o desenvolvedor pode especificar um conjunto de serviços Web a serem executados em paralelo e então, definir regras para selecionar, dentre eles, aqueles que devem consolidar (*commit*) ou serem compensados, depois do término de suas execuções.

O modelo de transações da WebTransact utiliza um critério de correção, denominado *2L-guaranteed-termination*, o qual define um conceito mais fraco de atomicidade que considera as necessidades do ambiente de serviços Web. Baseado neste critério, o modelo de transações da WebTransact define protocolos que garantem a execução *correta* e *segura* de composições de serviços Web. O conceito de execução correta significa que uma determina composição será executada de acordo com a sua especificação, i.e., a semântica transacional especificada pelo desenvolvedor é garantida pela infraestrutura, bem como a propriedade *2L-guaranteed-termination* [108]. O conceito de execução segura significa que a infraestrutura WebTransact somente executará especificações de composições cujos serviços Web suportem o comportamento transacional necessário, i.e., se existe a possibilidade de um determinado serviço Web ser compensado após a sua execução então, ele deve suportar compensação. Os conceitos de correção e segurança empregados pela WebTransact são derivados dos trabalhos nas áreas de modelos de transação estendidos (ETMS) ([30], [111]) e dos trabalhos na área de coordenação de processos transacionais ([1], [3], [114], [115]).

A infraestrutura WebTransact é um trabalho multidisciplinar relacionado a varias outras áreas tais como, composição de serviços eletrônicos (*e-service composition*), coordenação de processos transacionais (*transactional process coordination*), sistemas de gerenciamento de *workflow*, e sistemas de computação distribuída.

Existe uma quantidade expressiva de trabalhos na área de sistemas de computação distribuída ([7], [18], [25], [28], [30], [40], [43], [98]), coordenação de processos transacionais ([1], [3], [114], [115]), e sistemas de gerenciamento de *workflow*([44], [55],

[79], [90], [113], [146]). Estes projetos endereçam o problema relacionado ao suporte a transações distribuídas, contudo eles não consideram a coordenação de serviços com capacidades dissimilares e suportados por provedores de serviços autônomos. Considerando-se que serviços Web possuem capacidades dissimilares com relação ao seu comportamento transacional, esta característica mostra-se fundamental. Desta forma, a implementação de transações envolvendo serviços Web é consideravelmente mais complexa, quando comparada a um ambiente onde todos os componentes distribuídos suportam comportamento transacional uniforme. Outro aspecto a ser considerado é o fato destes trabalhos não terem sido desenvolvidos levando-se em conta os padrões baseados em XML que suportam a tecnologia de serviços Web.

Recentemente, sistemas voltados a composição de serviços eletrônicos tem atraído a atenção da academia ([21], [54]) e da indústria ([8], [22], [75], [130], [137]). Os trabalhos nesta área estão concentrados em definir primitivas para composição de serviços e automação da coordenação de serviços. A maior parte destes trabalhos considera os padrões XML para serviços Web. Contudo, as primitivas propostas para composição de serviços não tratam de forma direta os problemas associados à homogeneização de serviços Web. Ainda, o suporte a transações proposto nesta área não considera a mediação de serviços que ofereçam suporte transacional dissimilar.

Pelo que sabemos, não existem outros trabalhos propondo infraestruturas integradas que forneçam mecanismos que suportem a construção de composições de serviços Web escaláveis, manuteníveis, e robustas. A WebTransact endereça esta nova classe de interações transacionais envolvendo serviços Web, oferecidos por múltiplas organizações, através da mediação das capacidades dissimilares dos serviços Web, enquanto suporta a semântica de transações distribuídas envolvendo o mediador e os provedores de serviços Web.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho apresenta uma visão geral da infraestrutura WebTransact. Todos os detalhes referentes à WebTransact podem ser encontrados em [108].

O restante deste trabalho está organizado da seguinte forma.

No Capítulo 2, nós apresentamos uma visão geral da infraestrutura WebTransact. O objetivo deste capítulo é dar ao leitor uma visão unificada dos componentes da WebTransact. Primeiramente, nós descrevemos a arquitetura da WebTransact. Depois, descrevemos de forma genérica os principais componentes da arquitetura. Começamos descrevendo o serviço remoto, que é o componente responsável pela integração dos serviços Web. Então, apresentamos o serviço de mediador, que é o componente responsável pela camada que oferece a visão homogeneizada dos serviços Web. Depois, descrevemos como as composições de serviços Web são construídas utilizando-se da camada de serviços Web homogeneizados. Finalmente, apresentamos uma visão geral do modelo de execução utilizado pela infraestrutura WebTransact.

Após a apresentação da infraestrutura WebTransact, nós apresentamos as conclusões derivadas dos resultados do nosso trabalho e, em seguida, apresentamos as direções futuras de pesquisa.

2. Visão Geral da Infraestrutura WebTransact

Este Capítulo apresenta uma visão geral da infraestrutura WebTransact. Inicialmente, é apresentado um quadro geral de sua arquitetura. A seguir, são explicados sucintamente os componentes que fazem parte dessa arquitetura. Finalmente, é esboçada a semântica de operações da infraestrutura WebTransact. Todos os componentes, bem como a semântica operacional da WebTransact, são discutidos detalhadamente em [108].

2.1 ARQUITETURA

Conforme mostrado na Figura 2.1, a infra-estrutura WebTransact viabiliza a composição de serviços Web através da adoção de uma arquitetura com múltiplas camadas formadas por vários componentes especializados. Os programas de aplicação interagem com *composições de serviços de mediador* escritas por desenvolvedores de composições. Tais composições são definidas através de padrões de interação transacionais envolvendo os serviços de mediador. *Serviços de mediador* fornecem uma interface homogênea de (múltiplos) serviços remotos semanticamente equivalentes. *Serviços remotos* encapsulam a interface de um serviço Web específico bem como o seu comportamento, fornecendo as informações de mapeamento necessárias para converter mensagens do formato heterogêneo de serviço Web para o formato do mediador.

A arquitetura WebTransact encapsula o formato de mensagem, seu conteúdo e o comportamento transacional de múltiplos serviços Web, fornecendo diferentes níveis de serviços agregados. Primeiro, a arquitetura WebTransact fornece a funcionalidade de acesso uniforme a múltiplos serviços Web. Serviços remotos resolvem conflitos envolvendo as diferentes representações de informações usadas por serviços Web distintos, bem como conflitos devido a incompatibilidades de conteúdo de cada serviço Web. Além de resolver conflitos estruturais e de conteúdo, serviços remotos também fornecem informações sobre a interface e o comportamento transacional suportado pelos serviços Web. Em segundo lugar, a arquitetura WebTransact fornece uma camada de serviços remotos agregados. Serviços de mediador agregam serviços remotos semanticamente equivalentes fornecendo uma visão homogênea e integrada de múltiplos

serviços Web heterogêneos. Finalmente, padrões transacionais de interação são construídos sobre estes serviços de mediador.

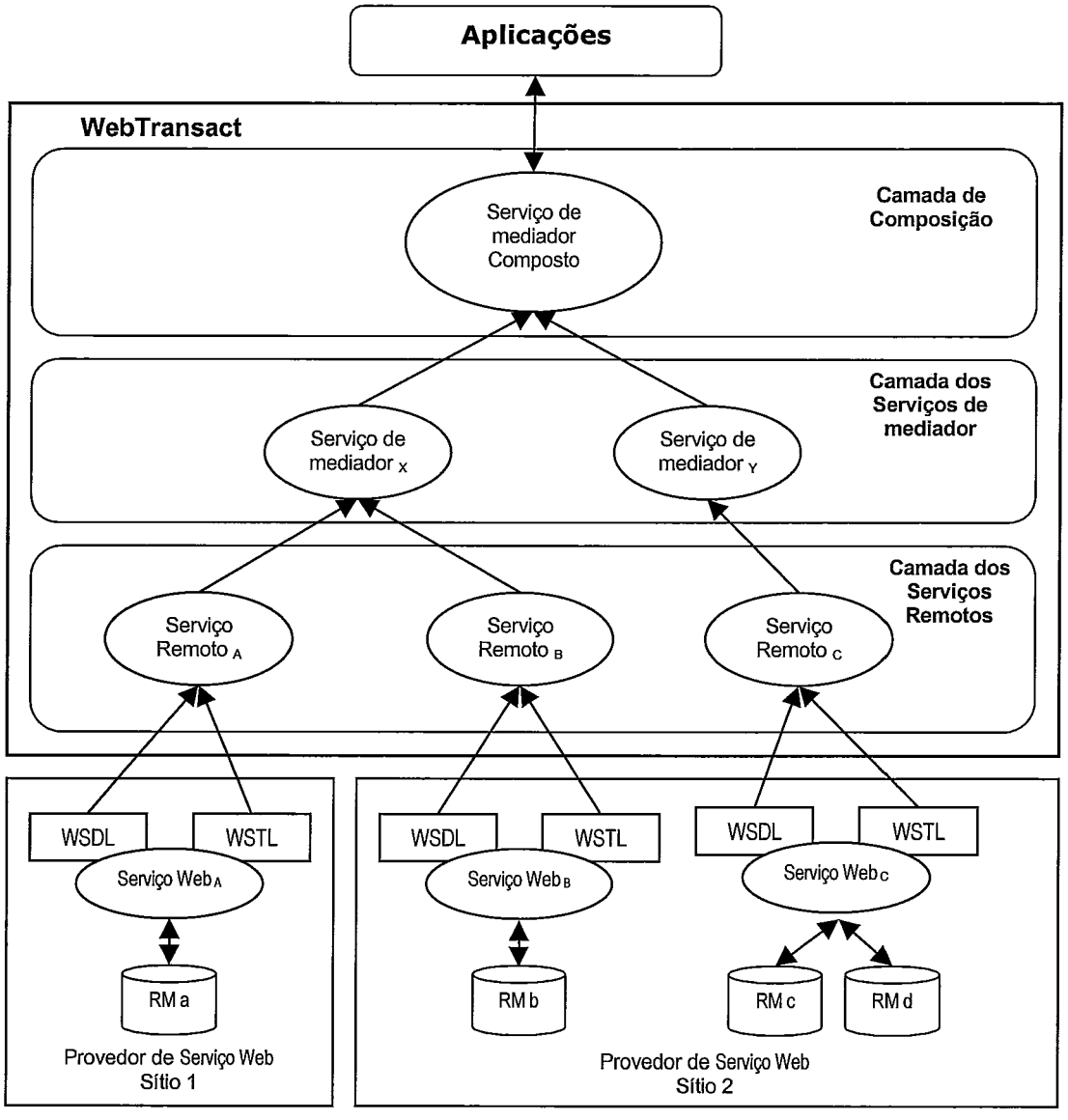

Figura 2.1 – A arquitetura da WebTransact.

Na WebTransact, a integração dos serviços Web se dá através de duas linguagens baseadas em XML: A *Web Service Description Language* (WSDL), que é a linguagem padrão para a descrição de interface de serviços Web, e a *Web Service Transaction Language* (WSTL) [108], que é a nossa proposta para viabilizar a composição transacional de serviços Web heterogêneos. WSTL é uma linguagem construída sobre a WSDL, estendendo-a com novas funcionalidades para suportar a composição de serviços

Web (Figura 2.2). Através da WSDL, os serviços remotos descobrem como interagir com os serviços Web. Através da WSTL, um serviço remoto sabe o suporte transacional oferecido pelo serviço Web. Além da descrição do comportamento transacional dos serviços Web, a WSTL também é utilizada para especificar todas as outras tarefas relacionadas à mediação de serviços na WebTransact. Estas tarefas incluem: especificação de mapeamentos para resolução de dissimilaridades de representação e de conteúdo, definição das interfaces dos serviços de mediador, e especificação de padrões de interação de composições.

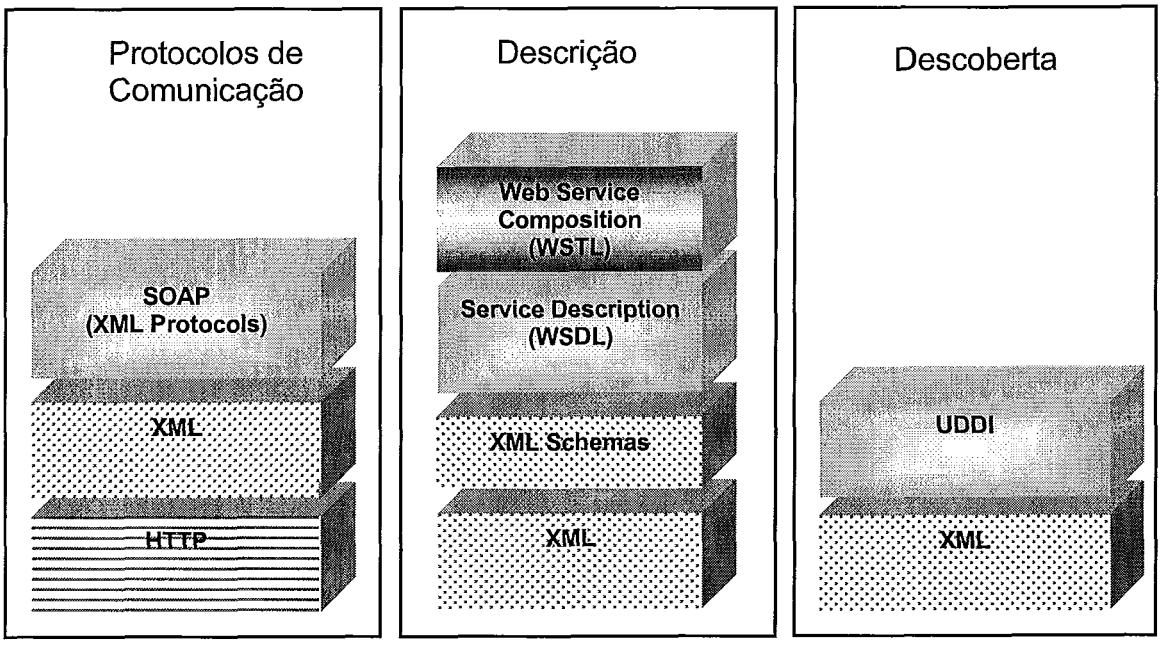

Figura 2.2– A relação da WSTL com os padrões XML da tecnologia de serviços Web.

A arquitetura distribuída da WebTransact separa a tarefa relacionada à agregação e homogeneização de serviços Web da tarefa de especificação de padrões de interação transacionais fornecendo, desta forma, um mecanismo genérico para tratar a complexidade introduzida pela existência de um grande número de serviços Web heterogêneos.

A arquitetura da infraestrutura WebTransact fornece funcionalidades específicas para o tratamento da complexidade inerente ao desenvolvimento de composições de serviços Web. Como os serviços do mediador fornecem uma visão homogênea dos

serviços Web, o desenvolvedor de composições não tem que se preocupar com a heterogeneidade ou a distribuição dos serviços Web. Ainda, a especificação de composições não referencia de forma direta as interfaces de serviços Web, desta forma, mudanças nestas interfaces não influenciam as especificações. Em um ambiente estritamente autônomo, esta característica é de fundamental importância. Outro aspecto relevante é a definição explícita da semântica transacional. Como os serviços web descrevem as suas políticas transacionais através de definições WSTL, a WebTransact pode coordenar a execução de composições de acordo com a política oferecida por cada serviço Web, gerando, desta forma, execuções robustas destas composições.

Nas próximas Seções, nós descrevemos de forma genérica os componentes da arquitetura WebTransact.

2.1.1 A Camada de Serviços Remotos

Cada *serviço remoto* representa uma unidade lógica de trabalho que acessa um conjunto de *operações remotas* implementadas por um determinado sítio (*site*) [108]. Cada operação remota possui uma assinatura e um *comportamento transacional* bem definido, o qual pode ser *compensable*, *retriable*, ou *pivot*, como no modelo de transações flexíveis [32]. Uma operação remota é *compensable* se, após a sua execução, os seus efeitos podem ser compensados através da execução de outra operação remota. Desta forma, para cada operação remota compensável, deve ser especificada qual é a operação remota que deve ser executada de forma a compensar os seus efeitos. Uma operação remota é *retriable* se existe a garantia de que ela terminará com sucesso após um número finito de sucessivas tentativas. Uma operação remota é considerada *pivot* se ela não for *compensable* nem, tampouco, *retriable*. Na WebTransact, os conceitos de operações remotas *pivot* e *compensable* são utilizados para garantir a terminação de composições. Provedores de serviços remotos podem não fornecer suporte a transações como, por exemplo, a interface 2PC. Neste caso, não é possível colocar a execução de uma operação remota em um estado similar ao estado de preparado-para-comitar (*prepared-to-commit*). Logo, após a execução de uma operação remota do tipo *pivot*, os seus efeitos tornam-se persistentes e, como não existe uma operação compensatória para operações *pivot*, não é possível desfazer os seus efeitos. Por esta razão, uma operação remota do tipo *pivot* só pode ser executada quando, após a sua execução, a composição alcançar um estado no qual ela se encontre pronta para terminar com sucesso. Neste estado, o sistema possui a

garantia de que não existirá a necessidade de desfazer nenhuma operação que por ventura já tenha sido comitada.

A Figura 2.3 mostra o diagrama de transição de estados de cada tipo de operação remota. Para as operações remotas do tipo *pivot*, somente as transições entre os estados *não-executado* e *executando* são controlados pela WebTransact, as outras transições de estado são controladas pelos sistemas remotos responsáveis pela execução da operação remota. Logo, após uma operação remota do tipo *pivot* atingir o estado *executando*, o sistema WebTransact não possui mais nenhum controle sobre as futuras transições de estado desta operação remota. Nas operações remotas de tipo *retriable*, o sistema WebTransact controla as transições entre os estados *não-executado* e *submetido* bem como as transições entre os estados *submetido* e *executando*. Como a transição entre os estados *submetido* e *executando* é controlada pelo sistema WebTransact, a terminação com sucesso de uma operação *retriable* é garantida. O diagrama de transições de estados das operações remotas do tipo *compensable* é similar ao diagrama das operações remotas do tipo *pivot*, com exceção da transição entre os estados *comitado* e *compensado*. Esta transição de estado é controlada pelo sistema WebTransact, desta forma é possível compensar os efeitos de uma operação remota do tipo *compensable*, caso seja necessário. A Figura 2.3 mostra ainda um outro tipo de operação remota, denominado *virtual-compensable*. Este tipo representa as operações remotas que suportam a interface padrão do protocolo *two-phase commit* (2PC). Estas operações são tratadas como operações do tipo *compensable*, mas, de fato, os seus efeitos não são compensados pela execução de outra operação, mas sim, eles esperam em um estado preparado-para-comitar até que a composição atinja um estado no qual é seguro comitar a operação remota. Desta forma, operações remotas do tipo *virtual-compensable* referenciam operações de serviços Web cujo sistema local possui (e expõe) algum tipo de serviço para coordenação de transações distribuídas.

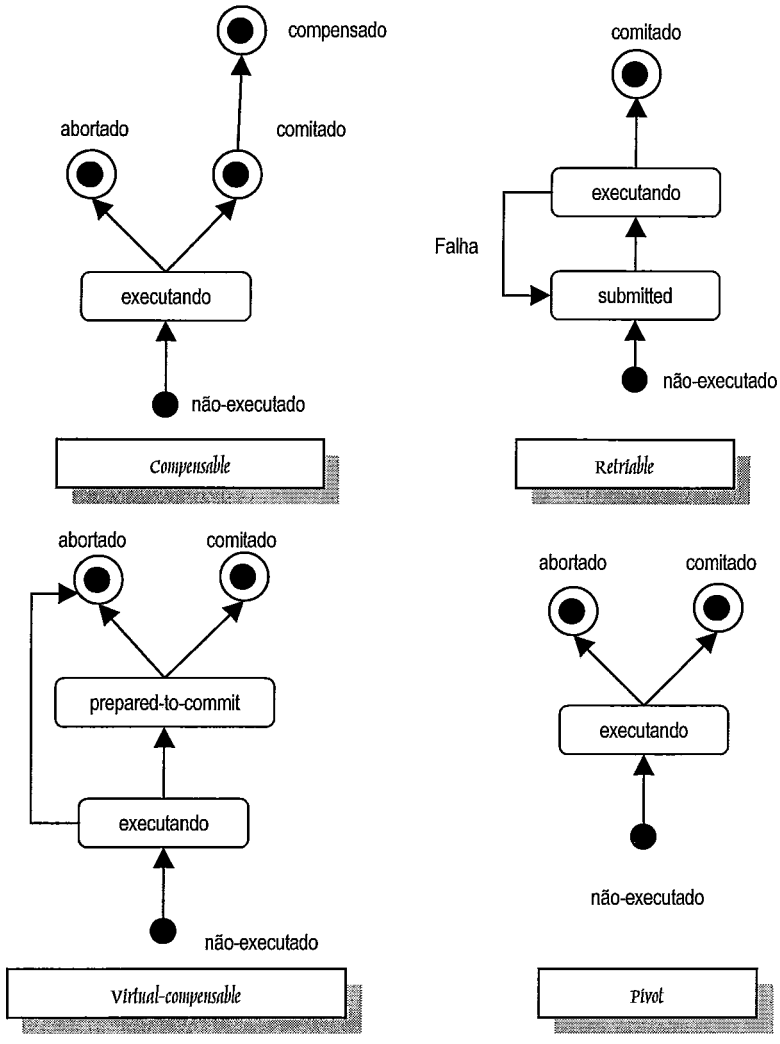

Figura 2.3 – Diagramas de transição de estado das operações remotas.

2.1.2 A Camada de Serviços de Mediador

Serviços de mediador agregam serviços remotos semanticamente equivalentes provendo, desta forma, uma visão homogênea de serviços remotos heterogêneos [108]. Diferentemente dos serviços remotos que são unidades lógicas de trabalho que executam operações remotas em um determinado sistema, serviços de mediador são serviços *virtuais* responsáveis pela delegação da execução de suas operações a um ou mais serviços remotos. Esta delegação é feita utilizando-se os serviços remotos (semanticamente equivalentes) agregados pelo serviço de mediador. Como ocorre com as operações remotas, operações de serviços de mediador têm um comportamento transacional bem definido que pode ser *compensable*, *retriable*, ou *pivot*.

O comportamento transacional de uma operação de um serviço de mediador é baseado no comportamento transacional de suas operações remotas agregadas. Quando todas as operações remotas agregadas por uma operação de um serviço de mediador possuem o mesmo tipo de comportamento transacional, por exemplo, *compensable*, então o comportamento transacional da operação do serviço de mediador terá o mesmo valor, i.e., *compensable*. Por outro lado, se a operação do serviço de mediador agregar operações remotas com comportamentos transacionais diferentes, então seu comportamento transacional será o comportamento transacional menos restritivo entre todos os comportamentos transacional de suas operações remotas agregadas (Figura 2.4). O comportamento transacional mais restritivo é o comportamento transacional *pivot*, enquanto o menos restritivo é o comportamento transacional *compensable*. Esta gradação entre comportamentos transacionais é usada para garantir a terminação segura de composições. Uma operação de um serviço de mediador que agregue pelo menos uma operação remota, cujo tipo seja *compensable*, pode participar em qualquer execução de composição. Por outro lado, uma operação de serviço de mediador, cujas operações remotas agregadas sejam todas do tipo *pivot*, pode participar de execuções de composições que solicitem esta operação após alcançarem um estado no qual a sua terminação com sucesso esteja garantida. Como o serviço de mediador delega a execução de suas operações a seus serviços remotos agregados, quando uma determinada operação deste serviço agrega somente operações remotas do tipo *pivot*, a execução de tal operação só pode ser delegada a uma operação remota do tipo *pivot*. Na Seção 2.1.1 vimos que depois que uma operação remota *pivot* é executada, a composição tem que entrar em um estado no qual esteja pronta para terminar com sucesso. Desta forma, operações de um serviço de mediador que agreguem somente operações de serviço remotos que sejam do tipo *pivot*, só podem participar em composições que solicitem esta operação quando a composição alcança um estado no qual ela está pronta para terminar com sucesso.

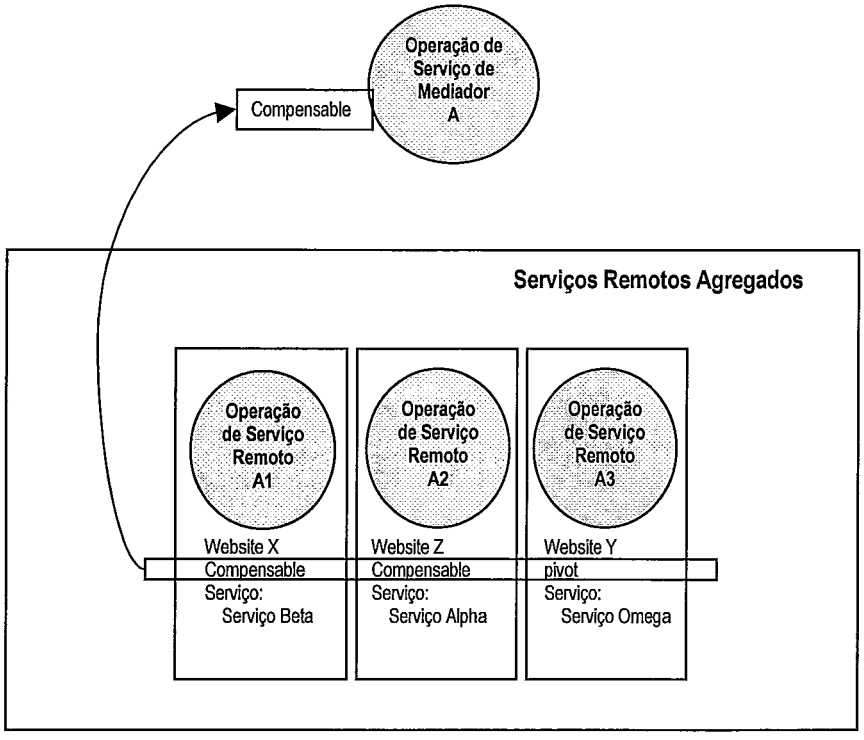

Figura 2.4 – Exemplo de comportamento transacional de serviços de mediador.

Cada tipo de operação de serviço de mediador tem o mesmo diagrama de transição de estado de seu tipo relacionado de operação remota (Figura 2.3), com exceção da ausência do tipo *virtual-compensable*. O comportamento transacional específico de operações remotas do tipo *virtual-compensable* é isolado pela operação de serviço de mediador que as agrega. Operações de serviço de mediador que agregam operações remotas do tipo *virtual-compensable* expõem a mesma interface que operações de serviço de mediador que agregam somente operações remotas que realmente sejam do tipo *compensable*. Este isolamento da semântica específica das operações remotas do tipo *virtual-compensable* simplifica o protocolo de coordenação de execução de composições, como pode ser verificado em [108].

2.1.2.1 Resolução de Dissimilaridades Semânticas e de Conteúdo de Serviços Web

Com o intuito de prover uma camada de serviços homogênea, cada serviço de mediador expõe uma interface única que é utilizada para especificar composições de serviços Web. Como serviços Web agregam serviços remotos semanticamente equivalentes, os quais possivelmente possuem diferentes interfaces, torna-se necessário

fornecer informações de mapeamento entre a interface suportada pelo mediador e cada uma das interfaces suportadas pelos seus serviços remotos agregados. Na WebTransact, cada *port type* WSDL é importado como um novo serviço remoto [108]. Como o elemento *port type* define a sintaxe para chamar um conjunto de operações remotas, i.e., uma interface suportada por determinado serviço Web, cada elemento *port type* é considerado como um serviço remoto independente. Um serviço remoto liga um serviço de mediador a um elemento WSDL *port type*, provendo informações de mapeamento entre as operações do serviço do mediador e as operações do *port type*. Ainda, o serviço remoto contém a descrição de conteúdo do serviço Web representado pelo *port type* em questão. A informação de mapeamento descreve como os parâmetros de entrada de uma operação remota são construídos a partir dos parâmetros de entrada da operação de serviço de mediador relacionada, bem como os parâmetros de saída e as mensagens de falha recebidas como resultado de uma chamada àquela operação remota são mapeados para as mensagens de saída ou falha da operação de serviço de mediador associada. A descrição de conteúdo especifica se um determinado serviço remoto possui a capacidade de responder à determinada solicitação. Por exemplo, considere os serviços remotos de reserva de carros rm_1 e rm_2. Considere ainda que o serviço remoto rm_1 pode fazer reservas de carros mundialmente, enquanto que o serviço remoto rm_2 aceita somente reservas de carros dentro do território brasileiro. Agora, considere que o serviço do mediador ms_1 agrega rm_1 e rm_2. Se ms_1 recebe uma solicitação para efetuar uma reserva de carro no território argentino então, ms_1 deverá delegar essa solicitação somente para rm_1. O serviço de mediador ms_1 sabe, consultando a descrição de conteúdo, que rm_2 não é capaz de responder a solicitações de reservas de carros fora do Brasil.

2.1.3 A Camada de Composição de Serviços de Mediador

Um *serviço de mediador composto* descreve padrões de interação transacionais de um conjunto de operações de serviços de mediador que cooperam entre si de modo a realizar uma determinada tarefa. Tal padrão de interação define o fluxo de execução das operações envolvidas como também o nível de atomicidade e robustez de uma determinada composição. Na WebTransact, uma composição é especificada usando elementos WSTL [108].

A WSTL modela composições como *tarefas compostas*. Uma tarefa composta é representada por um grafo direcionado rotulado no qual vértices representam passos de execução e as arestas representam o fluxo de controle e dados entre diferentes passos de execução. Cada passo de uma tarefa composta é uma *tarefa atômica* ou outra tarefa composta. Uma tarefa atômica é uma unidade de trabalho que é executada por uma operação de serviço de mediador. Então, cada tarefa atômica possui uma associação com uma operação de serviço de mediador, a qual é invocada quando a tarefa é executada.

Tarefas são identificadas por um nome e possuem uma assinatura, um conjunto de dependências de execução, um conjunto de ligações de dados, e, opcionalmente, um conjunto de regras.

A *assinatura* de uma tarefa atômica está relacionada às mensagens de entrada, saída, e falha da operação de serviço de mediador que é usada para implementar a tarefa. A *assinatura* de uma tarefa composta está relacionada às mensagens de entrada, saída, e falha de suas tarefas componentes. Uma tarefa componente pode ser uma tarefa atômica ou outra tarefa composta.

Dependências de execução são baseadas no estado de execução das tarefas componentes de uma determinada tarefa composta, definindo o primeiro tipo de arestas no gráfico que representa uma tarefa composta. Uma dependência de execução é definida entre tarefas relacionadas e define uma restrição na ocorrência temporal entre os eventos de início e término destas tarefas. Dependências de execução definem a ordem na qual devem ser executadas tarefas, i.e., o fluxo de controle de uma composição. Uma dependência de execução é sempre especificada com base no *estado de execução* de uma determinada tarefa componente.

Ligações de dados são mapeamentos entre mensagens que pertencem às assinaturas de tarefas relacionadas com a finalidade de permitir a troca de informação entre estas tarefas. Ligações de dados são o segundo tipo de arestas no gráfico que representa uma tarefa composta.

Regras especificam as condições sob as quais certos eventos acontecerão. As regras podem ser associadas tanto a dependências de execução ou a ligações de dados. Uma dependência de execução que possua uma regra associada será avaliada como verdadeira somente se a regra também for avaliada como verdadeira. Da mesma forma, uma ligação

de dados que possua uma regra associada, somente será avaliada como verdadeira se a sua regra associada for avaliada como verdadeira. Ligações de dados sem regras associadas, sempre são avaliadas como verdadeiras.

Além dos componentes descritos anteriormente, tarefas compostas podem possuir um conjunto de *tarefas obrigatórias*. Este conjunto especifica as tarefas, dentre aquelas que compõem a tarefa composta, que devem, obrigatoriamente, comitar para que a tarefa composta termine com sucesso. Um usuário pode especificar uma tarefa composta que agrega tarefas que são desejáveis, mas não essenciais, para a realização da tarefa em questão. O conjunto de tarefas obrigatórias permite a distinção entre tarefas *desejáveis* e *obrigatórias*, fornecendo mais flexibilidade para a especificação e execução de uma composição. Esta flexibilidade aumenta a robustez da composição, já que o conjunto de tarefas, as quais devem comitar para que a composição também comite, é formado somente por tarefas que são essenciais para realizar o objetivo de composição. Assim, a composição terminará com sucesso mesmo quando um subconjunto de suas tarefas componentes falhar, contanto que todas suas tarefas obrigatórias terminem com sucesso.

2.2 O MODELO DE EXECUÇÃO

Nesta Seção nós apresentamos uma visão geral da semântica operacional da WebTransact. Em [108], nós apresentamos os detalhes do modelo de transações da infraestrutura WebTransact, incluindo a sua descrição formal e a definição dos protocolos que suportam este modelo.

O modelo de execução da WebTransact possui dois níveis de controle. O primeiro é o nível de coordenação de tarefas compostas e o segundo é o nível de coordenação de tarefas atômicas. O primeiro nível especifica as regras utilizadas para interpretar uma especificação de tarefa composta. Tais regras asseguram que uma determinada tarefa composta será executada de acordo com a semântica de sua especificação e fornece garantias que tal execução será tratada como uma transação única. O segundo nível especifica as regras para coordenar a execução das operações de serviços de mediador que implementam tarefas atômicas. Este nível é responsável por prover uma interface transacional para as tarefas atômicas, a qual é utilizada pelo primeiro nível para a coordenação da execução de tarefa compostas. A coordenação de tarefas atômicas lida com problemas que surgem devido à relação um-para-muitos que existe entre operações

de serviço de mediador – que implementam as tarefas atômicas - e operações de serviço remoto – que são os provedores de serviço concretos, como também com os problemas que surgem devido ao comportamento transacional dissimilar de operações de serviço remotos semanticamente equivalentes.

A execução de uma tarefa composta consiste de duas fases distintas: a *fase de verificação*, e a *fase de escalonamento*. O objetivo da fase de verificação é assegurar a propriedade *garanteed-termination* de uma determinada instância de tarefa composta. Como as tarefas atômicas podem ser implementadas por operações de serviço de mediador que não são compensáveis, é necessário verificar se uma determinada instância de tarefa composta pode ser executada sem que haja o risco de efeitos parciais se tornarem persistentes. Na fase de verificação, todas as tarefas componentes são consultadas quanto ao comportamento transacional suportado, de modo a determinar a estrutura de execução resultante para a tarefa composta. Caso a estrutura resultante esteja de acordo com certas propriedades então, a execução é segura, i.e., ela assegura a propriedade *garanteed-termination*, e pode ser executada. Caso contrário, a instância não é segura e o sistema WebTransact envia uma mensagem de erro para o programa cliente. Instâncias de tarefas compostas que passam pela fase de verificação são os dados de entrada da fase de escalonamento. Nesta fase, a WebTransact escalona as tarefas componentes de uma determinada tarefa composta de acordo com a sua especificação.

O modelo de execução de tarefas atômicas define como uma chamada de uma determinada operação abstrata de serviço de mediador, na realidade, é executada por provedores de serviços remotos concretos. Instâncias de tarefas atômicas são submetidas para execução durante a fase de escalonamento de tarefas compostas. Este evento dispara a invocação da operação de serviço de mediador que implementa a tarefa atômica. Como a lógica de uma operação de serviço de mediador é implementada pelas suas operações remotas agregadas, quando tal operação é invocada, é necessário especificar como as suas operações remotas agregadas devem ser escalonadas. Isto é feito através de protocolos definidos pelo modelo de execução de tarefas atômicas.

3. Conclusões

A tecnologia de serviços Web fornece a infraestrutura para uma nova oportunidade de negócio onde uma empresa pode oferecer serviços agregados a seus clientes através da composição de serviços Web básicos. Contudo, a tecnologia de serviços Web atual provê mecanismos para solução *parcial* do problema de construir composições de serviços Web. A construção de composições de serviços Web robustas, manuteníveis, e escaláveis requer muito mais que somente a interoperabilidade entre programas cliente e serviços Web. Além da interoperabilidade, a construção de composições de serviços Web requer mecanismos para: descrição do suporte transacional dissimilar dos serviços Web, solucionar a heterogeneidade semântica e de conteúdo de serviços Web semanticamente equivalentes, especificação dos padrões de interação transacionais entre serviços Web, e coordenação de tais padrões de interação. Devido a este novo conjunto de requisitos, exigidos pelo ambiente de serviços Web, as infraestruturas para coordenação de processos de negócio atuais não podem ser diretamente aplicadas no desenvolvimento de composições de serviços Web. Logo, existe a necessidade de desenvolvimento de novas infraestruturas, especificamente desenvolvidas para endereçar os novos requisitos do ambiente de serviços Web. A contribuição principal deste trabalho é a especificação de uma destas infraestruturas, a infraestrutura *WebTransact*.

WebTransact endereça os requisitos necessários a construção de composições de serviços Web, robustas, manuteníveis e escaláveis. WebTransact trata o problema de construir composições de um modo integrado, provendo mecanismos para descrever o comportamento transacional dissimilar de serviços Web, para agregar serviços Web semanticamente equivalentes e para solucionar as suas heterogeneidades, para especificar padrões de interação de serviços Web que sejam seguros, e para coordenar tais padrões de interação de modo de transacional. De acordo com a pesquisa bibliográfica empreendida durante a confecção desta tese, não existem outros trabalhos na área de infraestruturas integradas que considerem todos os requisitos endereçados pela WebTransact.

3.1 CONTRIBUIÇÕES

A primeira contribuição deste trabalho é a descrição de uma visão integrada dos problemas relacionados à construção de composições de serviços Web. Poucos trabalhos ([8], [75], [130], [137]) abordam os problemas relacionados à construção de composições de serviços Web. Todos estes trabalhos concentram seus esforços em um subconjunto dos requisitos apresentados nesta tese. Por exemplo, o trabalho apresentado em [75] concentra-se no problema de definir padrões de interação de serviços Web mas não considera o problema de suportar Serviços Web com comportamento transacional dissimilar. Por outro lado, o trabalho apresentado em [130] contempla (parcialmente) este problema mas não considera um modelo de transação que apóie a sua proposta, tampouco considera o problema de agregar serviços Web semanticamente equivalentes. Nesta tese, nós apresentamos uma visão mais generalizada dos problemas relacionados à construção serviços complexos a partir da composição de serviços Web básicos. Tal generalização pode ser vista como um conjunto básico de requisitos necessários ao desenvolvimento de outras infraestruturas para composição de serviços Web.

A segunda contribuição é a definição de uma arquitetura de camadas múltiplas para tratar dos problemas de heterogeneidade e dinamismo existentes no ambiente de serviços Web. Esta arquitetura é baseada na tecnologia de mediadores ([18], [41], [49], [50], [74], [103], [129], [133], [135], [136], [151]). Porém, a arquitetura da WebTransact trata da mediação de serviços Web, enquanto que sistemas baseados em mediadores tratam da mediação de informação entre diferentes repositórios de dados. A arquitetura da WebTransact possui um conjunto de componentes especializados para efetuar a mediação de serviços Web. O componente *Serviço Remoto* soluciona conflitos que envolvem a representação dissimilar da informação de diferentes serviços Web bem como, os conflitos derivados da existência de serviços Web com diferentes conteúdos, mas semanticamente equivalentes. Além de solucionar conflitos estruturais e de conteúdo, serviços remotos provêem informação sobre a interface e o comportamento transacional suportado pelos Serviços Web. O componente *Serviço de mediador* agrega serviços remotos semanticamente equivalentes, fornecendo uma visão homogênea de serviços Web heterogêneos. Finalmente, composições de serviços Web são construídas sobre estes serviços de mediador. A arquitetura da WebTransact provê o isolamento necessário entre os provedores de serviços Web e as composições destes serviços. Como as composições são construídas utilizando-se as operações dos serviços de mediador, mudanças no

comportamento, conteúdo, ou interface dos serviços Web, bem como a agregação de um novo serviço Web, não influenciam de forma direta as composições já existentes. Esta camada de mediação provê um mecanismo bastante eficiente para tratar a heterogeneidade e dinamismo inerentes ao ambiente de serviços Web.

A terceira contribuição é a definição da *Web Service Transaction Language* (WSTL). A WSTL provê os elementos responsáveis pela definição do comportamento transacional suportado pelos serviços Web, pela solução das dissimilaridades semânticas e de conteúdo dos serviços Web, pela agregação de serviços Web semanticamente equivalentes, pela definição de interfaces de serviços de mediador, e pela definição de padrões de interação entre os serviços Web de uma dada composição. O elemento WSTL *transaction behavior* [108] permite a definição de diferentes níveis de suporte à transação, estes níveis variam de nenhum suporte à transação para sistemas que possuem suporte à coordenação de transações distribuídas. Desta forma, uma grande variedade de serviços Web, suportando diferentes tipos de interações transacionais, podem ser integrados à WebTransact. O elemento WSTL *remote service* [108] suporta a integração de serviços Web semanticamente equivalentes, fornecendo mecanismos para solucionar dissimilaridades semânticas e de conteúdo que ocorre nos serviços Web. O elemento WSTL *mediator service* [108] define uma visão homogênea de serviços heterogêneos, a qual é utilizada para construir composições, através da agregação de serviços Web semanticamente equivalentes. Os elementos WSTL: *composite task*, *atomic task*, *execution dependency*, *data link*, *rules*, and *mandatory tasks* [108] suportam a especificação de padrões de interação transacionais. A WSTL suporta a definição de tarefas compostas de forma recursiva assim, padrões de interação mais complexos podem ser definidos a partir de composições pré-existentes. O elemento *execution dependency* provê suporte para definir diferentes níveis de robustez e eficiência de composições de serviços Web. Através das dependências de execução *abort-start* [108], um desenvolvedor de composição pode definir caminhos de contingência para aumentar a robustez de uma determinada composição. Através do construtor de tarefas alternativas (*alternative constructor*) [108], um desenvolvedor de composição pode definir diferentes padrões de serviços concorrentes de modo a melhorar o tempo de resposta de uma determinada composição. Finalmente, a WSTL é uma extensão da *Web Service Description Language* (WSDL) [138], ou seja, ela é aderente a padrões XML que constituem a tecnologia de serviços Web. Devido a esta característica, a WSTL pode ser

usada como um padrão complementar para especificar composições transacionais de serviços Web. Assim, o programa cliente de composições escritas em WSTL pode ser um programa de aplicação ou qualquer outro sistema capaz de interagir com um serviço Web padrão.

A quarta contribuição deste trabalho é a definição de uma infraestrutura flexível para integrar serviços Web que possuem (e expõem) o suporte próprio para coordenação de transações distribuídas baseado no protocolo *two-phase commit* (2PC). Os conceitos fundamentais de tal infraestrutura são o uso do modelo *two-pipe* na comunicação de mensagens de sincronização de transação e o uso da infraestrutura WSDL para expor a interface 2PC para comunicação com um determinado gerente de transação local. A infraestrutura WebTransact faz uso do *Transaction Internet Protocol* (TIP) ([34], [35]) por coordenar as mensagens de sincronização de transação entre os gerentes de transação remotos e o sistema WebTransact. O protocolo TIP suporta o modelo *two-pipe*, satisfazendo a exigência fundamental para o processamento de transações no ambiente de serviços Web, ou seja, a separação das mensagens de sincronização de transação do protocolo de comunicação de aplicação. Com o intuito de aumentar a flexibilidade, a infraestrutura WebTransact não impõe nenhuma interface padrão para os comandos TIP. Um gerente de transação remoto pode expor sua interface TIP proprietária, contanto que seja definido um elemento WSTL *tmmap* [108] para aquela interface. Assim, qualquer gerente de transação que suporte o protocolo TIP (ou até mesmo qualquer variação do protocolo 2PC com falha presumida [91]) pode definir operações abstratas através de uma interface WSDL, mapear estas operações aos comandos TIP padrão, e então definir como estas operações abstratas devem ser enviadas usando as definições de ligação (*binding*) WSDL. Considerando que TIP e WSDL estão se tornando padrões bastante aceitos, esta infraestrutura combina ubiqüidade com flexibilidade.

A quinta contribuição desta tese é a definição e formalização de um modelo de transação e seus protocolos relacionados, para coordenação da execução de composições de serviços Web especificadas em WSTL. O modelo de transação da WebTransact explora o comportamento transacional dissimilar de serviços Web e garante a execução correta e segura de composições. A noção de correção de execução de composições é baseada nos critérios definidos pelo usuário (especificação da composição) e na propriedade *2L-guaranteed-termination*. A *2L-guaranteed-termination* é um novo critério de correção específico para o ambiente de serviços Web, onde a propriedade de

atomicidade da composição é relaxada para contemplar os conceito de operações compensáveis e não-obrigatórias e a potencial existência de múltiplos serviços Web semanticamente equivalentes.

3.2 TRABALHOS FUTUROS

A WebTransact pode ser usada como infraestrutura para outros trabalhos. Nos próximos parágrafos, nós apresentamos algumas das áreas de pesquisa que podem ser exploradas no contexto da infraestrutura WebTransact.

Arquiteturas de implementação: Experimentos interessantes podem ser feitos através da implementação (de partes) da infraestrutura WebTransact. Por exemplo, o repositório do mediador pode ser implementado por diferentes tipos de sistemas de banco de dados, como o relacional [27], objeto-relacional ([19], [122]), ou o orientado a objetos ([23], [66], [78]); outra opção interessante seria a utilização de um repositório XML nativo ([15], [24]). Os componentes de arquitetura WebTransact podem ser implementados usando plataformas distribuídas, como CORBA [97] e EJB [125]. A implementação destes componentes utilizando plataformas distribuídas favoreceria a flexibilidade e a escalabilidade do sistema WebTransact, já que tal implementação conduziria a uma arquitetura completamente distribuída e descentralizada. Como a WebTransact é aderente aos padrões da tecnologia de serviços Web, a interoperabilidade entre os componentes da WebTransact e os serviços Web pode ser implementada através de ferramentas comerciais de suporte a serviços Web existentes como, por exemplo, o "IBM's Web Services toolkit" [59] ou o ".Net framework SDK" [85]. O grupo de banco de dados do PESC/COPPE[3] já iniciou uma implementação da infraestrutura WebTransact utilizando o "IBM's Web Services toolkit". Este trabalho está sendo executado como parte de um projeto de tese de mestrado. O primeiro protótipo (com um conjunto limitado de funcionalidades) deve estar implementado até o final de setembro de 2002.

Coordenação de transações distribuída. Nós adotamos um modelo *two-pipe* juntamente com o protocolo de TIP para coordenar transações distribuídas que envolvam serviços Web que exponham a interface 2PC de seu coordenador local de transações distribuídas. Para facilitar a integração de diferentes sistemas de processamento de

[3] Programa de Engenharia de Sistemas e Computação (PESC) do Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia (COPPE).

transação é importante investigar quais seriam as dificuldades em se construir máquinas de estado TIP para os sistemas de processamento de transações distribuída mais comuns como o OMG's Object Transaction Service (OTS) [96], Sun's Java Transaction Service (JTS) [128], e o Microsoft's Distributed Transaction Coordinator (DTC) [86].

Descoberta dinâmica de serviços. Neste trabalho, nós não consideramos a descoberta dinâmica de serviços Web [65]. Os serviços Web são estaticamente integrados na WebTransact por um desenvolvedor que atua no papel de integrador de serviços Web. Porém, este não é um modo flexível para implementar a integração de serviços Web. Logo, existe a necessidade de se desenvolver ferramentas (semi-)automáticas para ajudar a tarefa de localizar e integrar serviços Web.

Ferramentas de apoio. A tarefa de construir composições através de elementos WSTL é um processo propenso a erros e tedioso. Então, existe a necessidade de se desenvolver ferramentas (gráficas) para especificar os fluxos de dados e controle de uma composição além de construir as mensagens de entrada, saída e de falha de tarefas compostas de forma automática. Tais ferramentas também devem prover mecanismos para verificação da correção das especificações de composições.

Segurança. Da mesma maneira que o comportamento transacional dos serviços Web, é provável que os requisitos relativos à segurança sejam dissimilares entre os serviços Web [116]. Logo, nós acreditamos que o comportamento específico relativo à segurança de serviços Web deve ser explicitamente definido, bem como devem ser desenvolvidos mecanismos por mediar tal comportamento dissimilar. Este é um tema interessante e importante a ser investigado no contexto da infraestrutura WebTransact.

Qualidade de Serviço. Da mesma maneira que aplicações em rede devem ser monitoradas para melhorar qualidade geral de sistema ([52], [148]), é necessário examinar e analisar as execuções de serviços Web para melhorar a qualidade e a eficiência das execuções de composições de serviços de mediador.

Domínios de aplicação. A WebTransact foi desenvolvida para servir como infraestrutura básica para construir processos de negócio utilizando Serviços Web. Porém, há outros domínios de aplicação onde as idéias da WebTransact podem ser úteis. Por exemplo, é importante investigar como a arquitetura de camadas múltiplas da WebTransact pode ser explorada para auxiliar no desenvolvimento de aplicações de

computação distribuídas específicas, como no domínio de *Grid computing* ([17], [73]). De acordo com [47], *Grid computing* provê mecanismos para integrar recursos geograficamente distribuídos e oferece acesso consistente e a baixo custo de recursos, independentemente do local físico destes recursos. A *Grid computing* permite o compartilhamento, seleção, e agregação de uma larga variedade de recursos computacionais geograficamente distribuídos (como supercomputadores, *clusters* de computadores, sistemas de armazenamento, e fontes de dados). Como a *Grid computing* está incorporando a tecnologia de serviços Web ([56], [120], [123]), nós acreditamos que o suporte à transações heterogêneas da WebTransact pode ser facilmente adaptado a plataformas *grid*.

Metodologias. A WebTransact provê as ferramentas necessárias para construção de um novo tipo de componente de software, à composição de serviços Web. Composições de serviços Web [102] têm uma granularidade maior bem como, carrega mais semântica que um componente servidor tradicional, como, por exemplo, um componente CORBA [97] ou um componente COM [84]. Logo, existe a necessidade de se investigar o impacto que as composições de serviços Web causarão no desenvolvimento de aplicações internet que utilizarem esta nova tecnologia ([26], [81], [80]).

4. Referências

[1] Agrawal, D., Abbadi, A. E., "Transaction Management in Database Systems". In: **[30]**, Chapter 1.

[2] Alonso, G., Fessler, A., Pardon, G., et al., "Correctness in General Configurations of Transactional Components". In: *Proceedings of the Symposium on Principles of Database Systems (PODS)*, Philadelphia, Pennsylvania, pp. 285-293, 1999.

[3] Alonso, G., Fessler, A., Pardon, G., et al., "Transactions in Stack, Fork, and Join Composite Systems". In: *Proceedings of the 7th International Conference on Database Theory (ICDT '99)*, Jerusalem, Israel, pp. 150-168, January 1999.

[4] Alonso, G., Fiedler, U., Hagen, C., et al., "WISE: Business to Business E-Commerce". In: *Proceedings of the International Workshop on Research Issues in Data Engineering (RIDE)*, Sydney, Australia, 1999.

[5] Alonso, G., Schuldt, H. , Schek, H.,. "Concurrency Control and Recovery in Transactional Process Management". In: *Proceedings of the Symposium on Principles of Database Systems in Philadelphia*, pp. 316-26, 1999.

[6] Andrade, J. M. , Carges, M. T. , MacBlane, M. R., "The TUXEDO System: An Open On-line Transaction Processing Environment". *Data Engineering Bulletin*, v. 17, n. 1, pp. 34-39, 1994.

[7] Ansari, M., Ness, L., Rusinkiewicz, M., et al., "Using Flexible Transactions to Support Multi-System Telecommunication Applications". In: *Proceedings of the 18th VLDB Conference*, August 1992.

[8] Arkin, A., "Business Process Modeling Language (BPML)". [http://www.bpmi.org/bpml-spec.esp], BPMI.org, March 2001.

[9] BEA Systems Press Release, "Web Services Architecture of BEA WebLogic E-Business Platform Enables Real Business-to-Business Transactions and Collaboration over the Internet". [http://www.bea.com/press/releases /2001/0226_web_services.shtml], February 2001.

[10] BEA, "BEA WebLogic Server, Programming WebLogic Enterprise JavaBeans". BEA Systems Inc., March 2001.

[11] Berners-Lee, T., Fielding, R., Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax". IETF RFC 2396, [http://ietf.org/rfc/ rfc2396.txt], August 1998.

[12] Bernstein, P. A., Goodman, N., "Concurrency Control in Distributed Database Systems", *ACM Computing Surveys*, v. 13, n. 2, pp. 185-221, 1981.

[13] Bernstein, P. A., Hadizlacos, V. , Goodman, N., *Concurrency Control and Recovery in Databases Systems*. Addison-Wesley, USA, 1987.

[14] Bernstein, P. A., Newcomer, E., *Principles of Transaction Processing for Systems Professionals*. Morgan Kaufmann, ISBN 1-55860-415-4, 1996.

[15] Bourret, R., "XML Database Products". [http://www.rpbourret.com/xml/ XMLDatabaseProds.htm], March 2002.

[16] Buchmann, A. P., Özsu, M. T., Georgakopoulos, D., et. al., "A Transaction Model for Active Distributed Object Systems". *Database Transaction Models for Advanced Applications*, Morgan Kaufmann, ISBN 1-55860-214-3, pp. 123-158, 1992.

[17] Buyya, R., Baker, M. (Eds.), *Grid Computing - GRID 2000, First IEEE/ACM International Workshop, Bangalore, India, December 17, 2000, Proceedings.* Lecture Notes in Computer Science, v. 1971, Springer Verlag, ISBN 3-540-41403-7, 2000.

[18] Carey, M. J., Haas, L. M., Schwarz, P. M., et al., *Towards Heterogeneous Multimedia Information Systems: The Garlic Approach.* Technical Report RJ 9911, IBM Almaden Research Center, USA, 1995.

[19] Carey, M. J., Mattos, N. M., Nori, A. K., "Object-relational database systems (tutorial): principles, products and challenges". In: *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, Tucson, Arizona, USA, 1997.

[20] Casanova, M. A., "The Concurrency Control Problem for Database Systems", *Lecture Notes in Computer Science*, v. 116, ISBN 3-540-10845-9, Springer 1981.

[21] Casati, F., Ilnicki, S., Jin, L., et al., "Adaptive and Dynamic Service Composition in eFlow". In: *Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CaiSE 2000)*, Stockholm, Sweden, pp. 13-31, June 2000.

[22] Casati, F., Shan, M., "Dynamic and adaptive composition of e-services", *Information Systems*, v. 26, n. 3, pp. 143-163, 2001.

[23] Cattel, R. G. G., Barry, D., *The Object Databases Standard 2.0.* Morgan Kaufmann, USA, 1997.

[24] Champion, M., "Storing XML in Databases", *eAI journal*, [http://www.eaijournal.com/PDF/StoringXMLChampion.pdf], pp. 53-55, October 2001.

[25] Chen, J., Bukhres, O. A., and Elmagarmid, A. K., "IPL: A Multidatabase Transaction Specification Language". In: *Proceedings of the 13th International Conference on Distributed Computing Systems*, Pittsburgh, PA, May 1993.

[26] Christophides, V., Hull, R., Karvounarakis, G., "Beyond Discrete E-Services: Composing Session-Oriented Services in Telecommunications". In: *Proceedings of the Second VLDB Workshop on Technologies for E-Services (TES 2001)*, Rome, Italy, pp. 58-73, September 2001.

[27] Codd, E. F., "A relational model of data for large shared data banks", *Communications of the ACM*, v. 13, n. 6, pp. 377-387, June 1970.

[28] Dayal, U., Hsu, M. and Ladin, R., "A Transactional Model for Long-Running Activities". In: *Proceedings of the 17th VLDB Conference*, September 1991.

[29] Dayal, U., Hsu, M., Ladin, R., "Organizing Long-Running Activities with Triggers and Transactions". In: *Proceedings of ACM SIGMOD Conf. on Management of Data*, 1990.

[30] Elmagarmid, A. K., *Database Transaction Models for Advanced Applications.* Morgan Kaufmann, 1992.

[31] Elmagarmid, A. K., Pu, C., eds., "Special Issue on Heterogeneous Databases", *ACM Comp. Surveys* v. 22, n. 3, 1990.

[32] Elmagarmid, A., Leu, Y., Litwin, W., et al., "A Multidatabase Transaction Model for InterBase". In: *Proceedings of the 16thVLDB Conference*, Brisbane, Australia, pp. 507-18, 1990.

[33] Eswaran, K. P., Gray, J., Lorie, R. A., et al., "The Notions of Consistency and Predicate Locks in a Database System", Communications of the ACM, v. 19, n.11, pp. 624-633, 1976.

[34] Evans, K., Klein, J. , Lyon, J., "Transaction Internet Protocol - Requirements and Supplemental Information". IETF RFC 2372, [http://ietf.org/rfc/rfc2371.txt], 1998.

[35] Evans, K., Klein, J. , Lyon, J., "Transaction Internet Protocol Version 3.0". IETF RFC 2372, [http://ietf.org/rfc/rfc2371.txt], 1998.

[36] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol - HTTP/1.1". IETF RFC 2616, [http://ietf.org/rfc/rfc2616.txt], 2000.

[37] Freed, N., Borenstein N., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies". IETF RFC 2045, [http://ietf.org/rfc/ rfc2045.txt], November 1996.

[38] Frolund, S., Govindarajan, K., "Transactional Conversations". In: *W3C Web services Workshop*, [http://www.w3.org/2001/03/WSWS-popa/paper50], San Jose, CA, April 2001.

[39] Garcia-Molina, H., Gawlick, D., Klein, J., et al., "Modeling Long-Running Activities as Nested Sagas", *Data Engineering Bulletin*, v. 14, n. 1, March 1991.

[40] Garcia-Molina, H., Gawlick, D., Klein, J., et al., *Coordinating Multi-transaction Activities*. Technical Report CS-TR-247-90, Princeton University, February 1990.

[41] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., et al., "The TSIMMIS Approach to Mediation: Data Models and Languages", *Journal of Intelligent Information Systems (JIIS)*, v. 8, n. 2, pp. 117-132, 1997.

[42] Garcia-Molina, H., Salem, K., "SAGAS". In: *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1987.

[43] Georgakopoulos, D., Hornick, M. F., Krychniak, P., et al., "Specification and Management of Extended Transactions in a Programmable Transaction Environment". In: *Proceedings of the Intl. Conf. on Data Engineering*, February 1994.

[44] Georgakopoulos, D., Hornick, M., Sheth, A., "An overview of workflow management: from process modeling to workflow automation infrastructure", *Intl. Journal on distributed and parallel databases*, v. 3, n. 2 , pp. 119-153, 1995.

[45] Gray, J., "Notes on database Operating Systems". In: *Operating Systems: An Advanced Course*, v. 60, Lecture Notes in Computer Science, Spring-Verlag, New York, 1978.

[46] Gray, J., Invited Paper, "The Transaction Concept: Virtues and Limitations". In: *Proceedings of the 7th International Conference on Very Large Data Bases*, pp. 144-154, September 1981.

[47] GRID Infoware, "Grid Computing Info Centre (GRID Infoware) - Home Page". [http://www.gridcomputing.com/], 2002.

[48] Gruhn, V., "Business Process Modeling and Workflow Management", *International Journal of Cooperative Information Systems IJCIS*, v. 4, n. 2-3, pp. 145-164, 1995.

[49] Gruser, J-R., Raschid, L., Vidal, M. E., et. al, "Wrapper Generation for Web Accessible Data Sources". *In: Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS)*, New York City, New York, USA, pp. 14-23, August 1998.

[50] Haas, L., Kossmann, D., Wimmers, E. L., et al., "Optimizing Queries across Diverse Data Sources". In: *Proceedings of the VLDB Conference*, 1997.

[51] Hadzilacos, V., "A theory of reliability in database systems", *Journal of the ACM*, v. 35, n. 1, pp. 121-145, 1988.

[52] Halteren, A., Fábián, G., Groeneveld, E., "Design and Evaluation of a QoS Provisioning Service". In: *Proceedings of the WG6.1 Third International Working Conference on Distributed Applications and Interoperable Systems (DAIS 2001)*, Kraków, Poland, pp. 189-202, September 2001.

[53] Härder, T. , Reuter, A., "Principles of Transaction-Oriented Database Recovery", *ACM Computing Surveys*, v. 15, n. 4, pp. 287-317, December 1983.

[54] Heuvel, W., Yang, J., Papazoglou, M. P., "Service Representation, Discovery, and Composition for E-marketplaces". In: *Proceedings of the 9th International Conference Cooperative Information Systems (CoopIS 2001)*, Trento, Italy, pp. 270-284, September 2001.

[55] Hsu, M., (ed.), *Special Issue on workflow and Extended Transaction Systems.* Bulletin of the Technical Committee on Data Engineering, v. 16, n.2, June 1993.

[56] Hyman, G., "IBM to Push Grid-Based Web Services ". [http://www.internetnews .com/ent-news/article/0,,7_977291,00.html], *InternetNews eletronic magazine*, April 2002.

[57] IBM White Paper, "The IBM WebSphere software platform and patterns for e-business - invaluable tools for IT architects of the new economy". [http://www4.ibm.com/software/info/websphere/ docs/wswhitepaper.pdf], 2000.

[58] IBM White Paper, "Web services by IBM". [http://www-3.ibm.com/software /solutions/webservices/overview.html], 2002.

[59] IBM White Paper, "Web services Toolkit". [http://www.alphaworks.ibm.com/ tech/webservicestoolkit], April 2002.

[60] IONA White paper, "IONA's Orbix E2A Web Services Integration Platform". [http://www.iona.com/products/webserv.htm], 2002.

[61] IONA, *Orbix TM - A Technical Overview.* Technical Report PN: PR-TEC-7-5, IONA Technologies Ltd. Dublin, Ireland, 1993.

[62] Jacobson, I., Invited Paper, "The Unified Process for Component-Based Development". In: *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE'99)*, Heidelberg, Germany, June 14-18, 1999.

[63] Jajodia, S., Kerschberg, L. (eds), *Advanced Transaction Models and Architectures*. Kluwer, ISBN 0-7923-9880-7, 1997.

[64] Jennings, N. R., Norman, T. J., Faratin, P., "ADEPT: An Agent-Based Approach to Business Process Management", *SIGMOD Record*, v. 27, n. 4, pp. 32-39, 1998.

[65] Karp, A. H, Smathers, K., "Advertising and Discovering Business Services". In: *W3C Web services Workshop*, [http://www.w3.org/2001/03/WSWS-popa/paper09], San Jose, CA, April 2001.

[66] Kim, W., *Modern Database System: The Object Model, Interoperability, and Beyond*. ACM Press and Addison-Wesley, 1995.

[67] Korth, H. F., Levy, E., Silberschatz, A., "A Formal Approach to Recovery by Compensating Transactions". In: *Proceedings of the 16th International Conference on VLDB*, 1990.

[68] Krychniak, P., Rusinkiewicz, M., Cichocki, A., et al., "Bounding the Effects of Compensation under Relaxed Multi-level Serializability", *Distributed and Parallel Databases*, v. 4, n. 4, 1996.

[69] Lampson, B. W., "Atomic Transactions". In: Goos, G., Hartmanis, J. (eds.), *Distributed Systems - Architecture and Implementation: An Advanced course*, Spring-Verlag, pp. 246-265, 1981.

[70] Lampson, B. W., Sturgis, H., *Crash Recovery in a Distributed Data Storage System*. Technical Report, Computer Science Laboratory, Xerox Palo Alto Research Center, Palo Alto, CA, 1976.

[71] Lampson, B. W. , Lomet, D. B., "A New Presumed Commit Optimization for Two Phase Commit". In: *Proceedings of the 19th International Conference on VLDB*, pp. 630-640, 1993.

[72] Lawrence, P., ed., *WfMC Workflow Handbook*. John Wiley & Sons Ltd., 1997.

[73] Lee, C. A. (Ed.), *Grid Computing - GRID 2001, Second International Workshop, Denver, CO, USA, November 12, 2001, Proceedings*. Lecture Notes in Computer Science, v. 2242, ISBN 3-540-42949-2, Springer 2001.

[74] Levy, A. Y., Rajaraman, A., Ordille, J. J., et al., "Querying Heterogeneous Information Sources Using Source Descriptions". In: *Proceedings of the VLDB Conference*, 1996.

[75] Leymann, F., "Web Services Flow Language (WSFL 1.0)". [http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf], May 2001.

[76] Liu, L., Pu, C., Lee, Y., "An Adaptive approach to query mediation across heterogeneous databases". In: *Proceedings of the Int. Conf. on Cooperative Information Systems*, IEEE Press, pp. 144-156, June 1996.

[77] Lynch, N. A., Merritt, M., Weihl, W. E., et al., *Atomic Transactions*. Morgan Kaufmann, ISBN 1-55860-104-X, 1993.

[78] Mauro, R. C. Mattoso. M. L. Q.,"Integração de LPOO e BDOO: Uma Experiência com JAVA e GOA++", In: *Proceedings of the XIII Simpósio Brasileiro de Banco de Dados*, Maringá, Brazil, October 1998.

[79] McCarthy, D., Sarin, S., "Workflow and Transaction in InConcert". In [55].

[80] Mecella, M., Pernici, B., "Designing Wrapper Components for E-Services in Integrating Heterogeneous Systems", *The VLDB Journal*, v. 10, n. 1, pp. 2-15, 2001.

[81] Mecella, M., Pernici, B., "Designing Components for e-Services". In: *Proceedings of the First VLDB Workshop on Technologies for E-Services (TES 2000)*, Rome, Italy, pp. 58-73, September 2001.

[82] Medina-Mora, R., Wong, H. and Flores, P., "ActionWorkflow TM as the Enterprise Integration Technology". In [55].

[83] Meyer, B., Mingins, C., "Component-Based Development: From Buzz to Spark - Guest Editors' Introduction", *IEEE Computer*, v. 32, n. 7, pp. 35-37, 1999.

[84] Microsoft Corporation and Digital Equipment Corporation, "The Component Object Model Specification". [http://www.opengroup.org/pubs/catalog/ax01.htm], October 1995.

[85] Microsoft Corporation, ".Net Framework - Home Page". [http://msdn.microsoft.com /netframework/default.asp], 2000.

[86] Microsoft Corporation, "Guide to Microsoft Distributed Transaction Coordinator". In: *Microsoft SQL Server 6.5 documentation*, Redmond, WA, 1996.

[87] Microsoft Corporation, "Microsoft Distributed Transaction Coordinator - Home Page". [http://msdn.microsoft.com/library/en-us/cossdk/htm/pgdtcintro_05ki.asp], 2002.

[88] Microsoft White Paper, "A Blueprint for Building Web Sites Using the Microsoft Windows DNA Platform". [http://www.microsoft.com/commerceserver/techres/whitepapers.asp], 2000.

[89] Microsoft White Paper, "A Platform for Web Services". [http://msdn.microsoft.com /library/en-us/dndotnet/html/Techmap_websvcs.asp], 2001.

[90] Miller, J. A, Palaniswami, D., Sheth, A. P., et al., "WebWork: METEOR$_2$'s Web-Based Workflow Management System", *Journal of Intelligent Information Systems*, v.10, n. 2, pp. 185-215, 1998.

[91] Mohan, C., Lindsay, B., Obermarck., R., "Transaction Management in the R* Distributed Database Management System", *ACM Transactions on Database Systems*, v. 11, n. 4, pp. 378-396, 1986.

[92] Mohan, C., "Transaction Processing and Distributed Computing in the Internet Age". In: *Proceeding of the International Conference on Extending Database Technology (EDBT)*, Tutorial Notes, Valencia, Spain, 1998.

[93] Moss, J. E. B., *Nested Transactions: An Approach to Reliable Distributed Computing*. Ph.D. thesis, MIT Press, Cambridge, MA, 1985.

[94] Navathe, S. B., Tanaka, A. K., Chakravarthy, S., "Active Database Modeling and Design Tools: Issues, Approache, and Architecture", *IEEE Data Engineering Bulletin*, v. 15, n. 1-4, pp. 6-9, 1992.

[95] Nielsen, H., Leach, P. , Lawrence, S., "An HTTP Extension Framework". IETF RFC 2774, [http://ietf.org/rfc/rfc2774.txt], 2000.

[96] OMG (Object Management Group). "Transaction Service Specificaiton". Revision 1.2.1, [http://www.omg.org/technology/documents/formal/transaction_service.htm], 2002.

[97] OMG (Object Management Group). The Common Object Request Broker: Architecture and Specification. Revision 2.0., July 1995.

[98] Özsu, M. T., Dayal, U., Valduriez, P., eds., *Distributed Object Management.* Morgan Kaufmann, San Mateo, CA, 1994.

[99] Papadimitriou, C. H., *The Theory of Database Concurrency Control*, Computer Science Press, 1986.

[100] Pardon, G., Alonso, G., "CheeTah: a Lightweight Transaction Server for Plug-and-Play Internet Data Management". In: *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt, pp. 210-219, September 2000.

[101] Paul, S., Park, E., Chaar, J., "RainMan: a Workflow System for the Internet". In: *Proceedings of USENIX Symp. on Internet Technologies and Systems*, 1997.

[102] Phillips, J., Foody, D., "Building a Fundation for Web Services", *eAI Journal*, [http://www.eaijournal.com/PDF/WSFoundationFoody.pdf], pp. 8-13, March 2002.

[103] Pires, P.F., Brügger, T., Mattoso, M. L. Q., "Mediators Metadata Management Services : An Implementation Using GOA++ System", *Electronic Journal of SADIO*, [http://www.dc.uba.ar/sadio/ejs], v. 2, n. 1, pp. 30-47, 1999.

[104] Pires, P.F., Mattoso, M. L. Q., "A CORBA Based Architecture for Heterogeneous Information Source Interoperability". In: *Proceedings of the 25th Technology of Object-Oriented Languages and Systems (TOOLS-25 '97)*, November, 1997, Melbourne Australia. IEEE Press, ISBN 0-8186-8485-2, June 1998.

[105] Pires, P.F., Benevides, M. R. F., Mattoso, M. L. Q., "Mechanisms for Specifying Communication Behavior in Object Oriented Database". In: *Proceedings of the 2000 ACM Symposium on Applied Computing*, Como, Italy, pp. 389-397, v. 1, March 2000.

[106] Pires, P.F., Mattoso, M. L. Q., "Aspectos de Implementação na Arquitetura Heterogênea HIMPAR [Interoperability Issues in the HIMPAR Heterogeneous Architecture]," Proc. of the XXI Brazilian Symp. Databases, São Carlos, Brazil, pp. 43-57, 1996 (in Portuguese).

[107] Pires, P.F., Raschid, L., "MedTransact: Transaction Support for Mediation with Remote Service Providers". In: *Proceedings of the 3rd International Conference on Telecommunications and Electronic Commerce*, Dallas, USA, November, 2000.

[108] Pires, P. F., Benevides, R. F. M., Mattoso, M., "WebTransact: A Framework for Specifying and Coordinating Reliable Web Service Compositions". In: *Technical Report ES-578/02, PESC/Coppe, Federal University of Rio de Janeiro,* [http://www.cos.ufrj.br /~pires/webTransact.html], April, 2002

[109] Pitoura, E., Bukhres, O. A. and Elmagarmid, A. K., "Object Orientation in Multidatabase Systems", *ACM Computing Surveys*, v. 27, n. 2, pp. 141-195, 1995.

[110] Pree, W., "Component-Based Software Development - A New Paradigm in Software Engineering?", *Software - Concepts and Tools*, v. 18, n. 4, pp. 169-174, 1997.

[111] Ramamritham, K., Chrysanthis, P. K., ed., "Advances in Concurrency Control and Transaction Processing", *IEEE Computer Society Press*, CA, 1997.

[112] Ranno, F., Shrivastava, S. K., Wheater, S. M., "A Language for Specifying the Composition of Reliable Distributed Applications". In: *Proceedings of the 18th Intl. Conf. on Distributed Computing Systems (ICDCS '98)*, Amsterdam, The Netherlands 1998.

[113] Rusinkiewicz, M., Sheth, A., "Specification and execution of transactional workflows". In: **[66]**, pp. 592-620.

[114] Schuldt, H., Alonso, G., Schek, H. J., "Concurrency Control and Recovery in Transactional Process Management". In: *Proceedings of the Symposium on Principles of Database Systems (PODS)*, Philadelphia, Pennsylvania, pp. 316-326, 1999.

[115] Schuldt, H., Schek, H. J., Alonso, G., "Transactional Coordination Agents for Composite Systems". In: *Proceedings of the International Database Engineering and Applications Symposium (IDEAS 1999)*, Montreal, Canada, pp. 321-331, August 1999.

[116] Seltzsam, S., Börzsönyi, S., Kemper, A., "Security for Distributed E-Service Composition". In: *Proceedings of the Second VLDB Workshop on Technologies for E-Services (TES 2001)*, Rome, Italy, pp. 147-162, September 2001.

[117] Shan, M., Davis, J., Du, W., et al., *HP Workflow Research: Past, Present, and Future*. Technical Report, Hewlett Packard Software Technology Laboratory, [http://www.hpl.hp.com/techreports/97/HPL-97-105.html], 1997.

[118] Sherman, M., "Architecture of the Encina Distributed Transaction Processing Family". In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, D.C., USA, pp. 460-463, May 1993.

[119] Short, S., "Building XML Web Services for the Microsoft® .NET Platform". ISBN 0-7356-1406-7, Microsoft Press, February 2002.

[120] Shread, P., "IBM To Announce Broad Support For Grid Computing, Globus". [http://www.internetnews.com/ent-news/article/0,,7_977291,00.html], *InternetNews eletronic magazine*, February 2002.

[121] Smerik, R., "TOP END: Distributed Transaction Processing for Open Systems". In: *Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems (PDIS 1993)*, San Diego, CA, USA, pp. 270-271, January 1993.

[122] Stonebraker, M., "Object-Relational Database Systems", *White-paper Montage Software Inc. (Now Called Illustra)*, 1994.

[123] Sullivan, T., "Sun combines grid computing, Web services". [http://www.infoworld.com/articles/hn/xml/02/02/15/020215hnsunonegrid.xml], *InforWorld eletronic magazine*, Feburary 2002.

[124] SUN Microsystems, "Developing Web Services with the Sun Open Net Environment". [http://www.sun.com/software/sunone/wp-spine/spine.pdf], 2002.

[125] SUN Microsystems, "Enterprise JavaBeans Specification 2.0. Sun Microsystems". [http://java.sun.com/products/ejb/docs.html], August 2001.

[126] SUN Microsystems, "Implementing Services on Demand and the Sun Open Net Environment (Sun ONE)". [http://www.sun.com/software/sunone/wp-implement/wp-implement.pdf], 2001.

[127] SUN Microsystems, "Java Remote Method Invocation (RMI)". [http://java.sun.com/ j2se/1.4/docs/guide/rmi/spec/rmiTOC.html], 2001.

[128] SUN Microsystems, "Java Transaction Service (JTS)". [http://java.sun.com/ products/jts/], 2002.

[129] Tanaka, A. K., Valduriez, P., "The Ecobase Project: Database and Web Technologies for Environmental Information Systems", *SIGMOD Record*, v. 30, n. 3, pp. 70-75, 2001.

[130] Thatte, S., "XLANG: Web Services for Business Process Design". [http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm], Microsoft Corporation, 2001.

[131] The Open Group, "Distributed Transaction Processing: XA+ Specification". [http://www.opengroup.org/dbiop/s423.pdf], Version 2, ISBN: 1-85912-046-6, June 1994.

[132] The Open Group, "X/Open Guide - Distributed Transaction Processing: Reference Model". Version 3, 1996.

[133] Tomasic, A., Raschid, L., Valduriez, P., "Scaling Access to Heterogeneous Data Sources with DISCO", *IEEE Transactions on Knowledge and Data Engineering*, v. 10, n. 5, pp. 808-823, 1998.

[134] UDDI.org, "UDDI Technical White Paper". [http://www.uddi.org/pubs/ Iru_UDDI_Technical_White_Paper.PDF], September 2000.

[135] Vassalos, V., Papakonstantinou, Y., "Describing and Using Query Capabilities of Heterogeneous Sources". In: *Proceedings of the VLDB Conference*, pp. 256-265, 1997.

[136] Vidal, M. E., Raschid, L., Gruser, J-R., "A Meta-Wrapper for Scaling up to Multiple Autonomous Distributed Information Sources". *In: Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS)*, New York City, New York, USA, pp. 148-157, August 1998.

[137] W3C (World Wide Web Consortium) Note, "Web Services Conversation Language (WSCL) 1.0". [http://www.w3.org/TR/2002/NOTE-wscl10-20020314/], March 2001.

[138] W3C (World Wide Web Consortium) Note, "Web Services Description Language (WSDL) 1.1". [http://www.w3.org/TR/2001/NOTE-wsdl-20010315], March 2001.

[139] W3C (World Wide Web Consortium) Recommendation, "Extensible Markup Language (XML) 1.0 (Second Edition)". [http://www.w3.org/TR/REC-xml], October 2000.

[140] W3C (World Wide Web Consortium) Recommendation, "HTML 4.01 Specification". [http://www.w3.org/TR/1999/REC-html401-19991224], December 1999.

[141] W3C (World Wide Web Consortium) Recommendation, "XML Path Language". [http://www.w3.org/TR/1999/REC-xpath-19991116], November 1999.

[142] W3C (World Wide Web Consortium) Recommendation, "XML Schema Part 1: Structures". [http://www.w3.org/TR/xmlschema-1/], May 2001.

[143] W3C (World Wide Web Consortium) Recommendation, "XML Schema Part 2: Datatypes". [http://www.w3.org/TR/xmlschema-2/], May 2001.

[144] W3C (World Wide Web Consortium) Recommendation, "XML Schema Part 0: Primer". [http://www.w3.org/TR/xmlschema-0/], May 2001.

[145] W3C (World Wide Web Consortium) Working Draft, "SOAP Version 1.2 Part 0: Primer". [http://www.w3.org/TR/soap12-part0/], December 2001.

[146] Wachter H., Reuter A., "The ConTract Model". In: **[30]**, Chapter 7.

[147] Weikum, G., Schek, H. J., "Concepts and Applications of Multilevel Transactions and Open Nested Transactions". In: [30] , Chapter 13.

[148] Widya, I., Stap, R. E., Teunissen, L. J. , et al., "On the End-User QoS-Awareness of a Distributed Service Environment". In: *Proceedings of the 6th International Conference on Protocols for Multimedia Systems (PROMS 2001)*, Enschede, The Netherlands, pp. 222-238, October 2001.

[149] Wiederhold, G., "Mediation in Information Systems", *ACM Computing Surveys*, v. 27, n. 2, pp. 265-267, 1995.

[150] Yeong, W. , Howes, T. , Kille, S., "Lightweight Directory Access Protocol". IETF RFC 2372, [http://ietf.org/rfc/rfc1777.txt], 1995.

[151] Zadorozhny, V., Raschid, L., Vidal, M. E., et. al, "Efficient Evaluation of Queries in a Mediator for WebSources". In: *Eletronic Proceedings of the ACM SIGMOD Conference*, [http://www.acm.org/sigs/sigmod/sigmod02/eproceedings/papers/Research-Zadorozhny-et-al.pdf], 2002.

[152] Zhang, A., Nodine, M. H., Bhargava B. K., et al., "Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems". In: *Proceedings of the ACM SIGMOD Conference*, pp. 67-78, 1994.