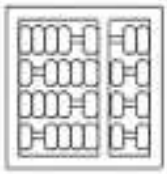

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Estudo sobre Propostas de Monitoramento em Serviços Web

A. F. Talon *M. B. F. Toledo*

Technical Report - IC-11-18 - Relatório Técnico

November - 2011 - Novembro

Estudo sobre Propostas de Monitoramento em Serviços Web

Anderson Francisco Talon[1], Maria Beatriz Felgar de Toledo[2]

[1] Instituto de Computação - Universidade Estadual de Campinas (UNICAMP)

13083-852 Campinas-SP, Brasil

talon@ic.unicamp.br

[2] Instituto de Computação - Universidade Estadual de Campinas (UNICAMP)

13083-852 Campinas-SP, Brasil

beatriz@ic.unicamp.br

Resumo. Nos últimos anos, a Computação Orientada a Serviços emergiu como um novo paradigma para o desenvolvimento de aplicações distribuídas. Neste paradigma, fornecedores de serviços desenvolvem seus serviços web e os publicam em repositórios de serviços. Os consumidores de serviços podem encontrar os serviços necessários nos repositórios e criar novos serviços a partir da composição de outros serviços. Mesmo que haja um acordo prévio entre as partes, muitos fatores como infra-estrutura de comunicação não confiável ou alteração dos serviços por parte dos fornecedores podem impedir que as condições pré-acordas sejam mantidas. Tendo em vista esse problema, é importante que os serviços sejam monitorados e comportamentos inconsistentes sejam detetados. O presente estudo tem como objetivo pesquisar o estado da arte em monitoramento de services web, além de fazer uma comparação destas propostas com relação aos tipos e formas de monitoramento presentes na literatura.

Palavras-Chave: Monitoramento; Serviços Web; WS-BPEL; Arquitetura Orientada a Serviços.

1. Introdução

Nos últimos anos, Arquitetura Orientada a Serviços (AOS) e Computação Orientada a Serviços (COS) emergiram como um paradigma para lidar com a complexidade de aplicações distribuídas. A idéia geral é baseada em padrões bem definidos e descrições de interface independentes da plataforma. Serviços web representam a mais comum utilização de AOS construída sobre os principais padrões SOAP e WSDL (Michlmayr et al., 2009).

AOS está se tornando rapidamente um paradigma importante para o desenvolvimento de aplicações distribuídas na era da internet. Neste paradigma, fornecedores de serviços desenvolvem seus serviços web de forma independente e os publicam nos repositórios de serviços baseados no padrão UDDI (OASIS, 2004). Os serviços são descritos na linguagem padrão WSDL (Chinnici et al., 2006). Os consumidores de serviços podem encontrar os serviços necessários nos repositórios e criar novos serviços a partir da composição de outros serviços. WS-BPEL representa hoje um padrão para a composição de serviços web (Wu et al., 2010).

Mesmo que haja um acordo prévio entre as partes, muitos fatores como infra-estrutura de comunicação não confiável ou alteração dos serviços por parte dos fornecedores podem impedir que as condições pré-acordas sejam mantidas. Estas incertezas exigem sistemas com técnicas de programação defensiva, como por exemplo, supervisão adequada da execução e flexibilidade na recuperação (Baresi & Guinea, 2008).

Uma solução para estas questões é o monitoramento contínuo dos serviços web em tempo de execução (Baresi & Guinea, 2005). O monitoramento em tempo de execução pode garantir o comportamento do serviço web monitorado. Os monitores operam em paralelo com o serviço monitorado a fim de detectar a inconsistência do comportamento. O comportamento desejado do serviço precisa estar na especificação do serviço e expressado em uma forma legível para a máquina (Bratanis et al., 2010).

2. Fundamentos

Nesta seção são apresentados os fundamentos relacionados com o Monitoramento em Serviços Web.

2.1. Serviços Web

Os serviços web emergiram como uma tecnologia promissora para a efetiva automação das interações inter-organizacionais (Alonso et al., 2010; Papazoglou, 2007).

Um serviço web é um tipo específico de serviço eletrônico que utiliza padrões abertos da internet para sua descrição, descoberta e invocação (Alonso et al., 2010). A tecnologia de serviços web tem vários padrões associados e descritos brevemente a seguir.

Descrição de Serviços: A descrição de serviços é baseada em interfaces, ela deve conter as operações oferecidas pelo serviço e o método de invocá-las. Além de especificar seu endereço através de um Identificador Uniforme de Recursos (URI) e o protocolo de transporte para invocar o serviço (Alonso et al., 2010). O padrão aceito atualmente para descrição de serviços é o WSDL (*Web Services Description Language*) (Chinnici et al., 2006).

Descoberta de Serviços: Após descrever o serviço deve-se disponibilizar esta descrição para que os interessados possam utilizá-lo. O conceito de diretório de serviços foi criado para armazenar descrições, nele é possível registrar e permitir que clientes procurem por serviços (Alonso et al., 2010). A especificação UDDI (*Universal Description, Discovery, and Integration*) (OASIS, 2004) define padrões para publicação e descoberta de serviços.

Interação de Serviços: Para se iniciar uma comunicação entre os serviços deve-se estabelecer primeiramente um protocolo de transporte, uma vez escolhido, a informação a ser trocada deve ser empacotada e formatada. O SOAP (*Simple Object Access Protocol*) (W3C, 2007) é um protocolo de comunicação capaz de encapsular mensagens escritas em

XML (*eXtensible Marckup Language*) (W3C, 2008) e trocar informações em uma plataforma distribuída.

Composição de Serviços: Um serviço web pode utilizar em sua implementação invocações a outros serviços disponibilizados por terceiros ou pela própria organização. A possibilidade de desenvolver serviços complexos a partir da invocação de serviços mais simples facilita a manutenção da aplicação, além de reduzir tempo e custo de desenvolvimento. Um processo de negócio pode ser constituído de vários serviços, porém precisa ser descrito por meio de uma linguagem interpretável por computador que seja capaz de organizar tais serviços de uma forma específica para atingir um objetivo de negócio (Weske, 2007). O padrão atual para a especificação de processos de negócio é a linguagem WS-BPEL (*Business Process Execution Language for Web Services*) (OASIS, 2007).

A utilização da tecnologia de serviços web permite diminuir o tempo de desenvolvimento de novos serviços devido a capacidade de reutilização e integração de sistemas legados desenvolvidos em plataformas distintas.

Outras informações que podem estar associadas a descrições de serviços são os atributos de qualidade do serviço (*QoS – Quality of Service*). Exemplos de atributos de QoS são: desempenho, disponibilidade, segurança, tempo de resposta, entre outros. O conjunto de QoS de um contrato eletrônico (Angelov & Grefen, 2008) costuma ser chamado de SLA (*Service Level Agreement*) (Keller & Ludwig, 2003; Sahai et al., 2002).

2.2. Monitoramento

Existem várias formas de classificar o tipo de monitoramento em serviços web. Nesta seção são apresentadas algumas formas de classificação.

Podemos classificar o monitoramento com relação à intrusão. Sendo assim, existem três grupos (Santos, 2011):

- *Monitoramento Intrusivo*: Consiste em adicionar a lógica de monitoramento dentro do código do processo de negócio (Baresi et al., 2004). Neste contexto, junto ao código BPEL são adicionadas diretivas de monitoramento. Estas diretivas são responsáveis por invocar o serviço de monitoramento, que por sua vez, valida se o fluxo do processo de negócio deve ou não ser interrompido ou alterado.
- *Monitoramento Levemente Intrusivo*: Nesse contexto seria alterar apenas o endereço de invocação do provedor no código do processo de negócio. Esse tipo de monitoramento é considerado levemente intrusivo, pois não adiciona nenhuma lógica de monitoramento no processo de negócio, no entanto, requer a alteração do endereço de invocação do provedor para um agente intermediário denominado monitor. Assim, todas as requisições do consumidor são realizadas para o monitor, que por sua vez valida as requisições de acordo com suas regras de monitoramento, e repassa tais requisições ao provedor. A resposta segue o fluxo oposto.
- *Monitoramento Não-Intrusivo*: Consiste em não alterar o código do processo de negócio que se deseja monitorar. Esse tipo de monitoramento é mais complexo, pois implica normalmente em implantar diretivas de monitoramento dentro da máquina de execução do processo de negócio. No caso de processos de negócio desenvolvidos na linguagem BPEL, é necessário implantar tais diretivas no servidor BPEL. Nesse caso o monitoramento se torna muito dependente da máquina BPEL: se a máquina for atualizada, o monitoramento pode tornar-se obsoleto. O monitoramento não intrusivo também pode ser realizado em nível de rede, ou seja, realizando a captura dos pacotes de comunicação entre o consumidor e provedor.

O monitor pode também ser classificado quanto à carga que suporta (Bratanis et al., 2010):

- *Monitor Heavy-Weight*: Um único monitor suporta o monitoramento de diferentes aspectos de um serviço web. Um interceptador de mensagens de requisição/resposta é usado para analisar os diferentes aspectos monitorados dentro do mesmo monitor. Entretanto, este tipo de monitor é mais difícil de implementar, porque ele é mais complexo. Além disto, uma falha no monitor implica na incapacidade de monitorar qualquer aspecto.

- *Monitor Light-Weight*: Um único monitor suporta o monitoramento de um aspecto de um serviço web. Vários monitores leves podem ser usados para monitorar diversos aspectos de um serviço. Embora estes monitores não sejam tão complexos de serem implementados, as mensagens de requisição/resposta precisam ser passadas para cada monitor. Entretanto, a falha de um monitor tem um impacto apenas nos aspectos monitorados, sendo que o resto dos monitores não são afetados.

O monitoramento pode ser feito em quatro níveis (Isozaki et al., 2008):

- "*Monitoramento no nível de sites*" é o nível mais simples de monitoramento, onde o alvo são as mensagens entre os sites. Não consegue monitorar o comportamento de cada instância de processo.
- "*Monitoramento no nível de processo*" realiza o monitoramento das mensagens entre as instâncias do processo.
- "*Monitoramento no nível de mensagens entre atividades*" realiza o monitoramento das mensagens entre as atividades. Este nível de monitoramento pode acompanhar a cooperação das mensagens entre instâncias de processos e o comportamento das mensagens das atividades de cada instância de processo. Analisando o resultado do monitoramento com a descrição do processo é possível monitorar o progresso nas mensagens das atividades de cada instância de processo.
- "*Monitoramento no nível de atividade*" realiza o monitoramento de todas as atividades. Este nível de monitoramento pode acompanhar a cooperação das mensagens entre as instâncias dos processos e o comportamento de todas as atividades de cada instância do processo.

Além das classificações apresentadas, outro aspecto a considerar nas arquiteturas de monitoramento é a forma de interceptação das mensagens. Existem três formas de interceptar as mensagens de requisição/resposta entre o fornecedor e o consumidor (Bratanis et al., 2010):

- *Interceptação baseada em Handler*: Um *handler* é associado ao serviço monitorado. As mensagens são encaminhadas primeiro para o *handler*, portanto ele é capaz de

interceptá-las antes de chegar ao serviço monitorado e ao consumidor, respectivamente.

- *Interceptação baseada em Wrapper*: O serviço monitorado é encapsulado dentro de outro serviço. O serviço resultante tem a mesma interface do serviço monitorado. Assim é possível interceptar as mensagens trocadas entre o serviço monitorado e o consumidor.
- *Interceptação baseada em Proxy*: Um nó intermediário atua como um proxy de rede. O proxy é capaz de interceptar as mensagens que passam pelo protocolo de transporte, antes de chegarem ao seu destino.

Outra forma de classificar os serviços de monitoramento pode ser com relação ao tempo em que o monitoramento acontece. Com relação ao tempo, podemos classificar as abordagens em dois grupos:

- *Síncrono*: O serviço é bloqueado enquanto o monitoramento é realizado.
- *Assíncrono*: O monitoramento é feito durante a execução do serviço sem que ele seja bloqueado.

O monitoramento muitas vezes é utilizado para uma recuperação do sistema, auxiliando em sistemas de auto-recuperação. Podemos destacar duas técnicas de recuperação (Baresi & Guinea, 2008):

- *Técnicas de backward* consistem em estratégias como "*rollback*" e "*compensation*", que permitem aos sistemas de restauração voltar para estados íntegros anteriores ao estado atual.
- *Técnicas de forward* tentam colocar o processo em condição de continuar, mas não necessariamente em um estado anterior válido de execução.

3. Propostas

Nesta seção são apresentadas as propostas presentes na literatura sobre o Monitoramento em Serviços Web.

WS-QoS

A proposta de Tian et al. (2004) permite uma seleção dinâmica de serviço de acordo com a QoS. Na inicialização, a aplicação cliente cria uma instância do *WS-QoS Requirement Manager* e do *WS-QoS Broker* (WSB). Antes da invocação do serviço, a aplicação cliente fornece seus requisitos de QoS para o *Requirement Manager* e então consulta o WSB sobre ofertas disponíveis que cumprem estes requisitos. O WSB seleciona a oferta mais adequada para o cliente do seu banco de dados local. Podemos observar esta seleção dinâmina de serviço na Figura 3.1.

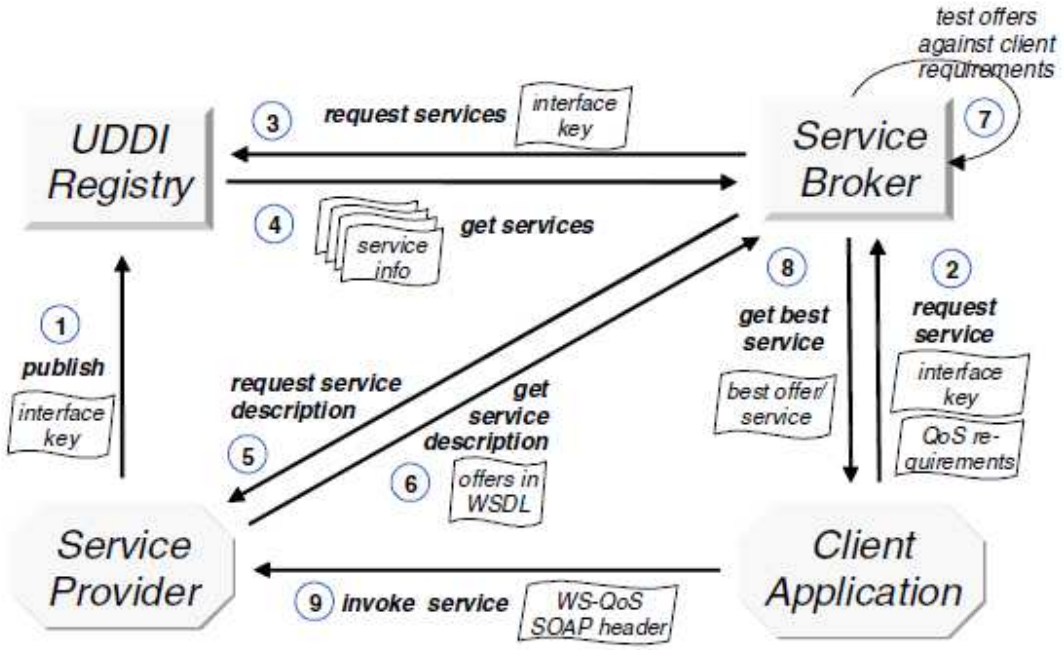

Figura 3.1: Interação entre os papeis.

O *WS-QoS Editor* permite tanto para o cliente do serviço como para o fornecedor do serviço uma forma fácil de editar as necessidades e ofertas de QoS por meio de uma interface gráfica.

O *WS-QoS Broker (WSB)* detém as informações atualizadas sobre as ofertas disponíveis para serviços solicitados recentemente. A primeira vez que um serviço web é requisitado por um cliente, um ou mais UDDI associados com o WSB são consultados. Os arquivos WSDL para estes serviços são verificados e uma lista de ofertas é construída. Para manter a lista de ofertas atualizada, O WSB consulta o UDDI periodicamente. Se uma oferta expira, ela é excluída do WSB. Quando o aplicativo cliente consulta o WSB sobre a oferta mais barata, ele envia seus requisitos de QoS como parte do pedido. Na ordem de preço, o WSB testa as ofertas disponíveis e se elas atendem os requisitos do cliente. A primeira oferta compatível é retornada para o cliente.

O *WS-QoS Monitor* exibe todas as ofertas disponíveis e os requisitos do cliente atual, tornando possível verificar o cumprimento de ofertas. Se nenhuma oferta apropriada pode ser encontrada, o conjunto de ofertas possíveis ajuda o usuário a avaliar quais os requisitos podem ser inadequados e fazer o ajuste necessário, a fim de fazer sua aplicação funcionar.

Além disto, o *WS-QoS Requirement Manager* pode ser configurado para registrar as exigências atuais. Uma vez que este arquivo é registrado no monitor, os requisitos podem ser visualizados na janela de observação de requisitos ou diretamente na janela de oferta do GUI.

Trabaho de Baresi & Guinea (2005)

Neste trabalho de Baresi & Guinea (2005), as regras de monitoramento são integradas aos processos WS-BPEL em tempo de instalação. A definição explícita e externa das regras de monitoramento permite uma boa separação entre as regras de negócio e as regras de controle. Estas regras possuem parâmetros que permitem uma adaptação em tempo de execução do grau de controle de monitoramento. Diferentes regras de monitoramento (e/ou parâmetros de monitoramento) podem ser associadas ao mesmo processo WS-BPEL,

permitindo que o usuário configure o grau de controle para uma execução específica, sem alterar o processo de negócio. Além disso, uma boa separação das regras de monitoramento é uma maneira eficaz de equilibrar o monitoramento e o desempenho do sistema.

A arquitetura da abordagem de monitoramento pode ser observada na Figura 3.2. O pré-processador BPEL^2 da figura analisa as regras de monitoramento e adiciona atividades ao código BPEL podendo associar pré ou pós-condições a invocações de serviços.

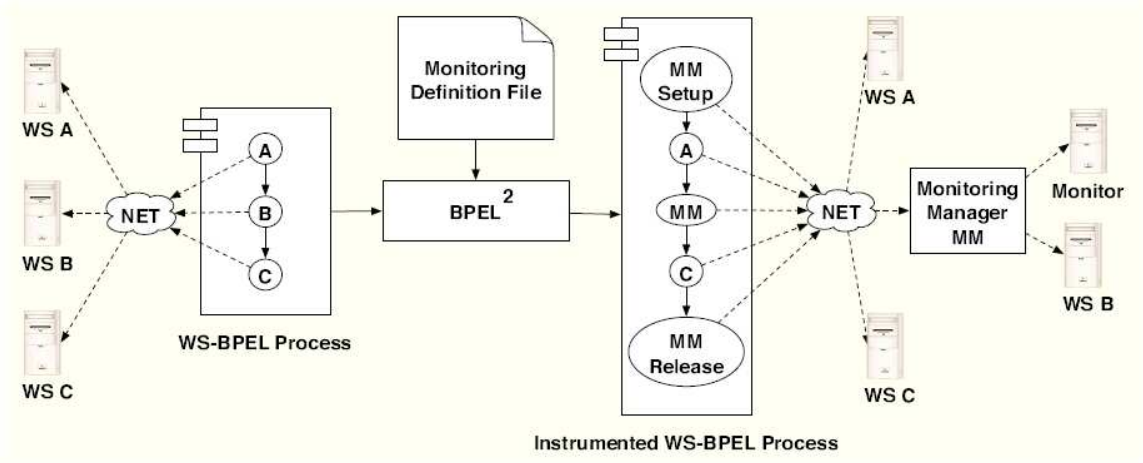

Figura 3.2: Abordagem de monitoramento.

O componente principal da abordagem é o *Monitoring Manager* (Figura 3.3) que é uma solução baseada em proxy para o monitoramento dinâmico. O *Manager* é capaz de interpretar as regras de monitoramento, de rastrear a configuração de um processo, de interagir com os coletores de dados externos e obter dados adicionais para um monitoramento específico, e de invocar serviços externos.

No caso de pré-condições, o *Manager* verifica se a regra precisa ser avaliada observando os parâmetros de monitoramento. Se as condições são verificadas corretamente, é invocado o serviço web original. Para as pós-condições, o processo é similar.

O *Manager* mantém uma tabela de configuração de cada processo em execução. Estas configurações são gerenciadas pelo *Configuration Manager*. Em especial, o *Manager* precisa saber: a configuração inicial do processo, as regras de monitoramento, e toda informação necessária para interagir com os serviços externos. A maioria destas

informações são enviadas para o *Manager* no começo do processo. Toda informação enviada durante a etapa de configuração é armazenada no *Configuration Manager* e associada com a execução de um processo através de um identificador único gerado pela máquina WS-BPEL. De forma similar, no final da execução do processo o *Manager* é avisado para liberar a configuração armazenada.

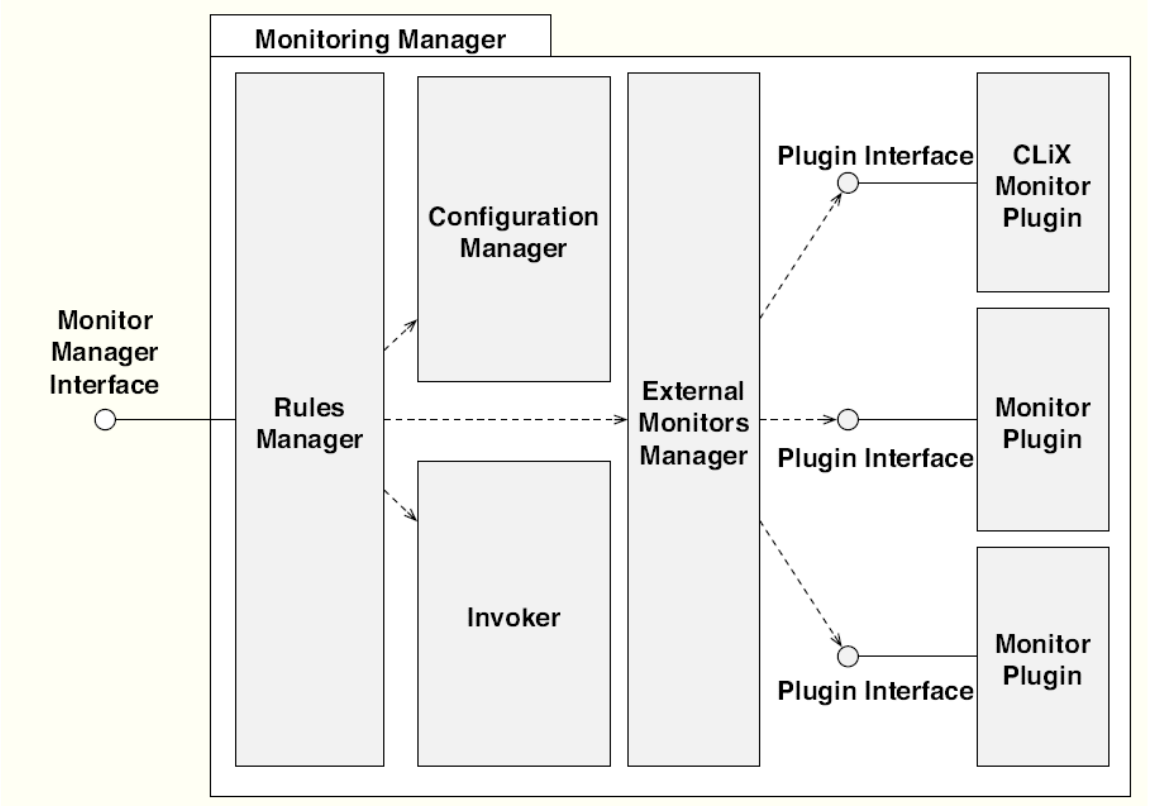

Figura 3.3: Arquitetura do gerente de monitoramento.

O *Manager* também disponibiliza uma interface gráfica para o usuário. Isto permite em tempo de execução consultar e modificar os valores da tabela de configuração. Por exemplo, é possível alterar o nível de prioridade do processo que está executando ou disponibilizar uma nova lista de provedores certificados associados com uma regra de monitoramento.

Os passos da interação dos componentes que cooperam durante a execução de um serviço são apresentados na Figura 3.4 para pós-condições. Inicialmente, o processo WS-BPEL envia os dados necessários para o *Manager* (Passo 1). Já que pré-condições não precisam

ser verificadas, o *Rules Manager* pede para o *Invoker* chamar o serviço externo (Passos 2 e 3). Quando o *Rules Manager* recebe o resultado do serviço chamado (Passos 4 e 5), ele interage com o *Configuration Manager* para determinar qual regra de monitoramento precisa ser verificada (neste caso, pós-condição) (Passo 6).

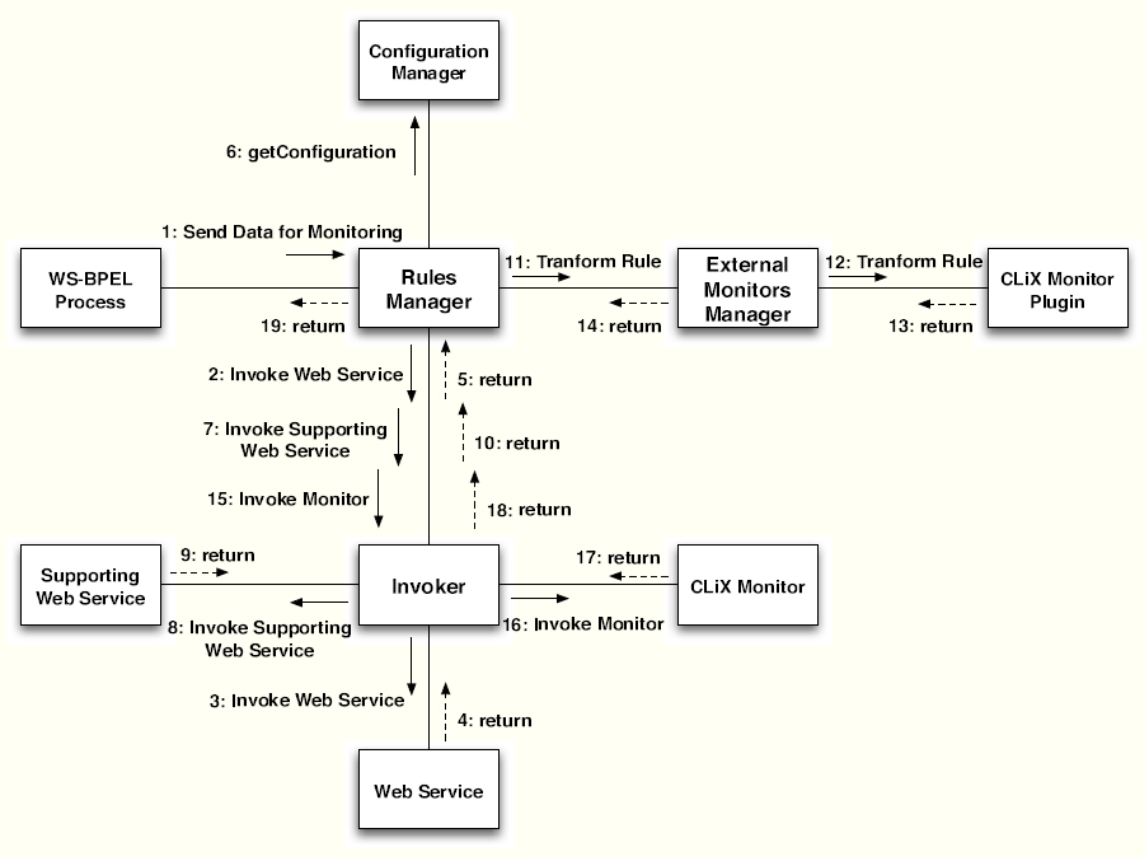

Figura 3.4: Interação dos componentes no gerente de monitoramento.

Examinando os parâmetros da regra, o *Rules Manager* decide dinamicamente se a regra precisa ser verificada ou não. Se a prioridade da regra for maior que a prioridade do processo, a regra deverá ser verificada. Então, o *Rules Manager* decide que dados adicionais serão necessários a partir dos coletores de dados externos. Neste caso, ele chama o *Invoker* para obtê-los (Passo 7). O *Invoker* pode chamar qualquer serviço desde que saiba: a URL do WSDL do serviço a ser chamado, o nome da operação a ser chamada, a lista das chaves para mapear os valores de entrada em partes de mensagem definidas na descrição WSDL, os valores de entrada do serviço a ser chamado, e a lista das chaves que indicam o

que é necessário receber nos valores de saída. O *Invoker* também pode ser chamado quando uma expressão usa uma WS-CoL para obter dados de monitoramento adicional a partir dos coletores de dados externos. Neste caso, a lista de chaves de saída é reduzida a somente uma chave que corresponde a mensagem de saída descrita na descrição da WSDL do serviço. Uma vez que os dados forem obtidos (Passos 8, 9 e 10), o *Rules Manager* começa a interação com *External Monitors Manager* (Passo 11). Este componente é responsável por gerenciar os diferentes tipos de monitores externos que o *Manager* pode trabalhar. Em especial, ele gerencia um conjunto de plugins que contém a lógica necessária para converter a sintaxe WS-CoL usada nas expressões de monitoramento na sintaxe necessária para utilizar em cada monitor externo. O plugin monitor também prepara os dados que precisam ser enviados para o monitor em um formato que o monitor consiga entender (Passo 12).

Quando os *External Monitors Manager* terminam (Passos 13 e 14), o *Invoker* é chamado novamente para invocar o monitor externo (Passo 15). Se o monitor responde com erro (Passos 16, 17 e 18), significa que a condição não foi satisfeita, o *Rules Manager* avisa o processo WS-BPEL retornando uma mensagem de erro padrão, como presente na descrição do WSDL. Se o monitor responde que a condição foi satisfeita (Passos 16, 17 e 18), o *Manager* pode continuar e retorna a resposta do serviço original para o processo WS-BPEL (Passo 19).

Dynamo

Dynamo (Dynamic Monitoring) é uma solução inovadora para a atual tecnologia BPEL com capacidades de auto-recuperação. O sistema foi apresentado nos trabalhos de Baresi & Guinea (2007) e Baresi et al. (2007). O sistema verifica se as expectativas funcionais e não-funcionais são atendidas, e em caso de anomalia, cria estratégias de recuperação. A implementação baseada em proxy é apresentada na Figura 3.5.

Dynamo propõe a utilização de regras de supervisão para restringir a execução de processos de negócio e atuar em comportamentos inesperados. Estas regras de supervisão ficam separadas da lógica de negócio e não interferem com a execução dos processos.

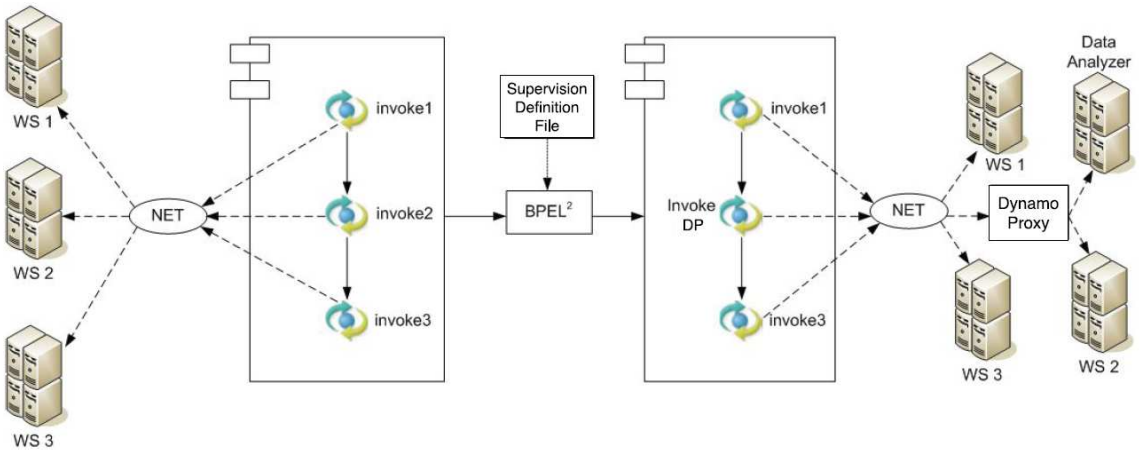

Figura 3.5: Implementação baseada em proxy.

Para a recuperação, Dynamo provê uma solução programável, flexível e extensível. A abordagem permite criar estratégias mais complexas através de uma combinação de ações mais simples.

Trabalho de Baresi & Guinea (2008)

O framework de supervisão proposto por Baresi & Guinea (2008) é composto por três componentes: (1) a máquina de execução que seleciona e executa regras de supervisão; (2) o subsistema de monitoramento; e, (3) o gerente de recuperação. A arquitetura (Figura 3.6) é composta por outros componentes, sendo os mais importantes o *Dynamic Invoker* para invocar dinamicamente serviços web, o *Configuration Manager* para armazenar regras de supervisão, e o *Storage* para armazenar variáveis históricas.

Uma regra de supervisão consiste de uma localização, conjunto de parâmetros (por exemplo, prioridade), uma propriedade de monitoramento especificada em WSCoL e um conjunto de estratégias de recuperação em WSReL.

WS-CoL (*Web Service Constraint Language*) é uma linguagem baseada em XML que define relacionamentos entre três tipos de variáveis: internas, externas e históricas.

WS-ReL (*Web Service Reaction Language*) é uma linguagem para definição de estratégias de reação. A recuperação é uma atividade consequente da descoberta de qualquer anomalia

no monitoramento. Assim como o monitoramento, a recuperação é feita de forma síncrona enquanto o processo está momentaneamente bloqueado. Assim que uma anomalia é descoberta, existem várias estratégias de recuperação. Cada estratégia é acompanhada por uma condição (escrita em WS-CoL), para indicar qual estratégia tem prioridade sobre a outra. As estratégias são definidas como um conjunto de etapas, que têm como objetivo a recuperação de uma anomalia. Se uma etapa falha, a próxima é considerada, e assim por diante.

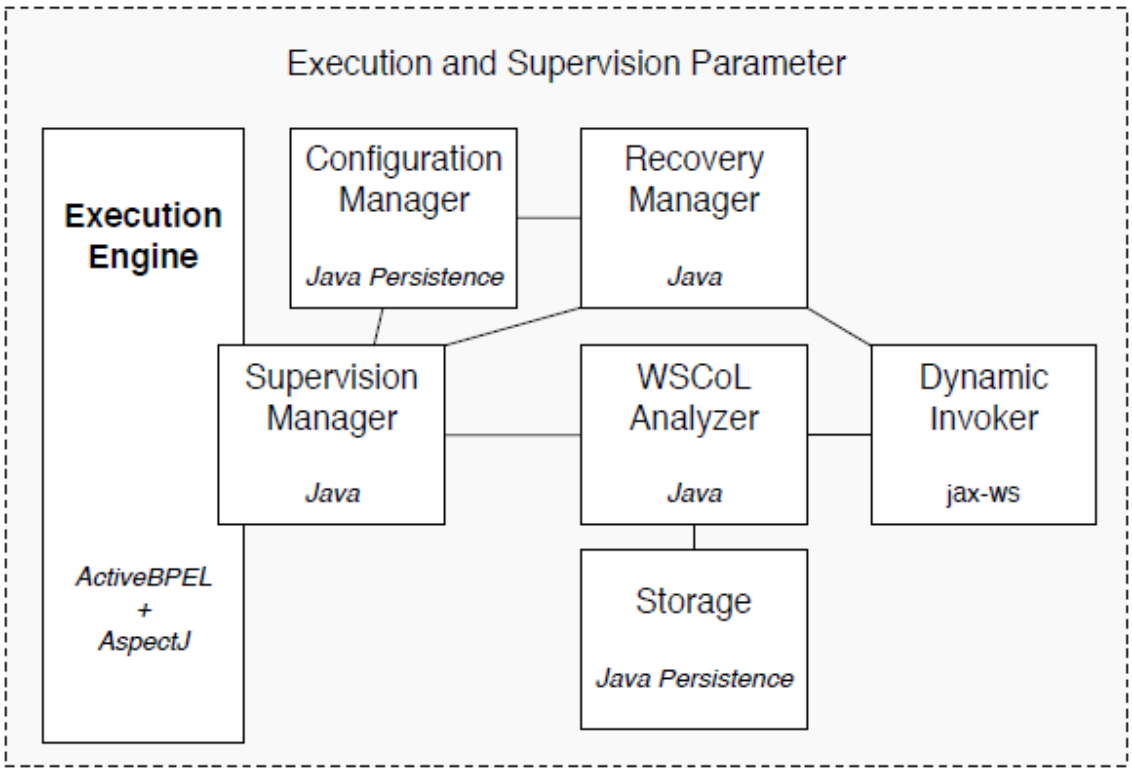

Figura 3.6: Framework de desenvolvimento e supervisão.

A máquina de execução é modificada para serem introduzidos monitoramento e recuperação. É criada uma representação em árvore do processo em execução. Na árvore, cada nó representa uma atividade do processo e tem a informação necessária para o desenvolvimento da atividade relacionada ao nó. São monitoradas as atividades que acontecem entre os processos e o mundo externo, portanto as atividades interceptadas são *Invoke*, *Receive* e *Pick*.

Assim que uma atividade termina, o controle é passado para o *Supervision Manager*. Este componente procura uma regra de supervisão definida no *Configuration Manager* que verifica se uma regra precisa ser executada ou se pode ser ignorada. Se a regra tem prioridade maior ou igual ao do processo, a regra precisa ser executada. Regras com prioridade "super alta" nunca podem ser ignoradas. A partir deste ponto o monitoramento começa. Primeiramente o *Supervision Manager* procura pelas variáveis internas. A variável é coletada diretamente do processo. Os dados coletados são enviados para o *WSCoL Analyser* junto com a propriedade de monitoramento. O *Analyser* coleta as variáveis externas através do *Dynamic Invoker* e as variáveis históricas através do *Storage Component*. A análise é feita e o resultado é passado para o *Supervision Manager*. Se a pós-condição está correta, o processo continua sua execução. Caso contrário, se uma anomalia é descoberta, as estratégias de recuperação expressas em *WSReL*, os dados coletados no monitoramento, e os dados necessários para chamar o serviço supervisionado, são passados para o *Recovery Manager*.

O *Recovery Manager* tem uma representação interna das estratégias de recuperação. Quando ele é inicializado, recebe todas as definições da recuperação, contendo as condições e as estratégias. As etapas da estratégia são inseridas em uma pilha de ações a serem tomadas. O framework utiliza *forward recovery*.

BP-Mon

O sistema BP-Mon (*Business Process Monitoring*), proposto por Beeri et al. (2007a, 2007b, 2008), é uma parte do BP-Suite, um novo conjunto de ferramentas baseadas no padrão BPEL. BP-Suite oferece uma interface uniforme, amigável e baseada em consultas que combina a análise das especificações dos processos, monitoramento do comportamento em tempo de execução, e análise de relatórios para uma compreensão do gerenciamento de processos. BP-Suite é composto por três subsistemas fortemente acoplados: BP-QL permite a consulta e a análise de especificações de processos; BP-Mon permite o monitoramento da execução das instâncias do processo; e BP-Ex permite uma análise posterior dos registros de sua execução. Os três sub-sistemas são todos baseados na mesma linguagem de consulta

simples, intuitiva e gráfica, cuja GUI é muito similar aos sistemas comerciais usados para o projeto de processos BPEL. Esta é uma característica importante do sistema, já que permite (a) o aprendizado rápido da linguagem e (b) formulação simultânea, pelo projetista do processo, tanto na especificação do processo quanto na verificação/monitoramento/análise das consultas sobre o processo.

As principais características do sistema são descritas a seguir.

Linguagem de Consulta: O sistema é baseado em uma linguagem de consulta gráfica que permite uma simples descrição da execução dos padrões a serem monitorados.

Otimização e Avaliação de Consulta: Os usuários devem ser notificados assim que um determinado padrão acontecer. Um algoritmo baseado em autômato suprime atividades de monitoramento redundantes.

Implementação e Instalação: Para suportar uma implantação flexível, o sistema BP-Mon compila uma consulta *q* dentro da especificação *S* do processo BPEL, cujas instâncias executam uma tarefa de monitoramento. Como para todas as especificações BPEL, *S* pode agora ser compilada automaticamente em código executável para executar no mesmo servidor de aplicação BPEL dos processos monitorados.

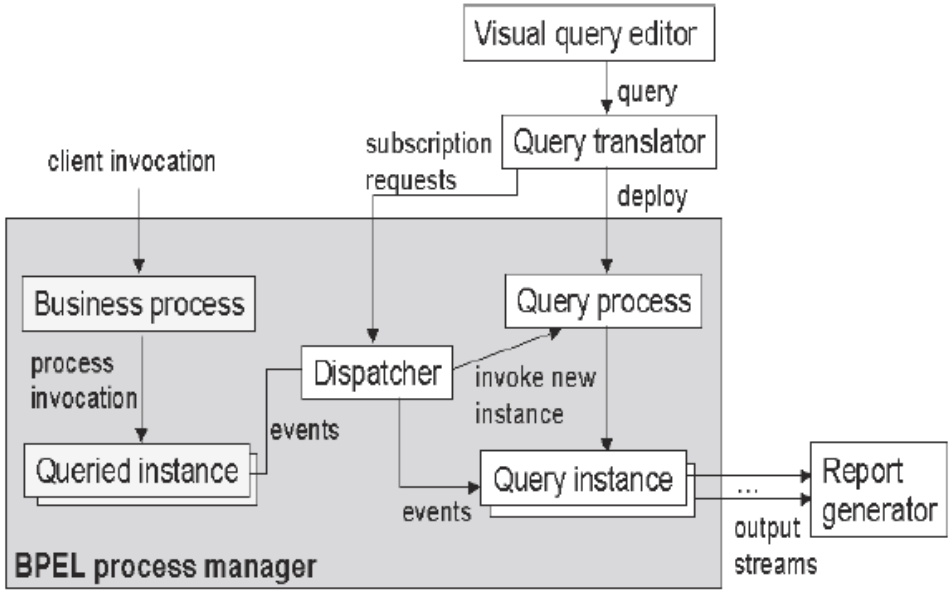

Figura 3.7: Arquitetura do BP-Mon.

Os componetes do sistema (Figura 3.7) são descritos a seguir.

Visual Editor. As consultas do BP-Mon são escritas por meio de um editor visual.

Query Translator. Para ter um suporte de implantação flexível, o sistema compila as consultas BP-Mon em especificações BPEL.

Dispatcher. O módulo despachante é responsável em tempo de execução pelo mapeamento entre os eventos das instâncias do processo e os processos de consulta.

Report Generator. O sucesso para um padrão de consulta está associado com a geração de relatórios ou ações corretivas.

Trabalho de Bianculli & Ghezzi

A arquitetura proposta por Bianculli & Ghezzi (2007) adota a abordagem AOP (*Aspect-Oriented Programming* (Kiczales, 1996)). Os aspectos são dinamicamente inseridos na máquina BPEL para monitorar a composição de serviços.

Esta solução permite que a lógica de negócio seja mantida separada da lógica de monitoramento, para conseguir uma boa modularização de código. O monitoramento é adicionado ao processo de negócio de forma transparente, ou seja, sem modificar a estrutura do processo, podendo ser ativado/desativado dinamicamente. Esta solução representa uma alternativa para as arquiteturas nas quais um proxy central se torna um gargalo. Além disto, elimina a necessidade de modificar o processo BPEL original para rotear suas invocações de serviços externos para o proxy.

A solução estende a implementação padrão da máquina ActiveBPEL com três componentes adicionais, como pode ser observado na Figura 3.8. Os componentes principais da arquitetura são descritos a seguir.

Main Interceptor: Ele intercepta e modifica a execução do processo dentro da máquina, em alguns *pointcuts*, usando AOP.

Specifications Registry: Ele contém as especificações contra as quais os serviços serão verificados com suas conformidades.

Monitor: O verificador de conformidade. O componente *Monitor* é na verdade um avaliador das especificações algébricas. Ele contém um gerador de estado simbólico (*symbolic state generator*) e um interpretador (interpreter), como pode ser observado na Figura 3.9.

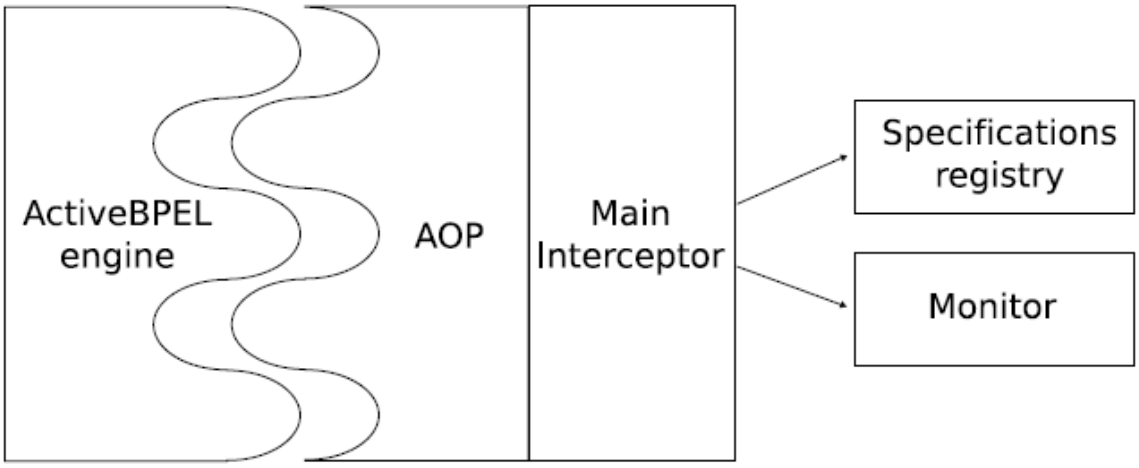

Figura 3.8: Monitoramento incorporado dentro da máquina BPEL.

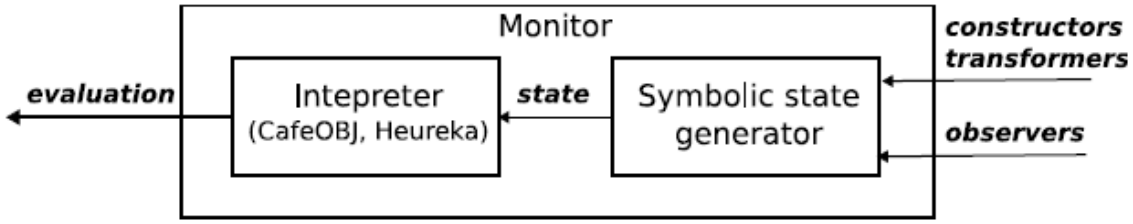

Figura 3.9: Arquitetura do monitor.

O gerador de estado simbólico mantém uma descrição legível da máquina de estado de cada descrição da especificação algébrica, para cada instância de processo. O estado é determinado pela sequência de operações construtor e transformador realizadas no serviço web correspondente. O estado é passado para o interpretador quando uma operação de observação é invocada, o interpretador então retorna uma constante, resultante da avaliação do estado, isto é, a partir da redução do termo equivalente da representação simbólica do estado.

Trabalho de Baresi et al. (2008a)

Neste framework proposto por Baresi et al. (2008a), existe uma separação clara entre as atividades de coleta de dados, monitoramento e recuperação. Entre as informações coletadas estão: os dados internos, que contêm o estado interno do processo; os dados externos, que fornecem informações do ambiente; e, os dados históricos, que representam as informações de execuções passadas. Abordagens diferentes de monitoramento são capazes de compartilhar os resultados parciais e colaborarem para uma avaliação mais completa. O framework pode acionar ações corretivas na mesma instância do processo, em instâncias diferentes (de diferentes processos), e também na definição do processo. É permitida uma interação entre as ações síncronas e assíncronas com um mecanismo de resolução de conflitos associado a uma prioridade. A Figura 3.10 apresenta a arquitetura do framework.

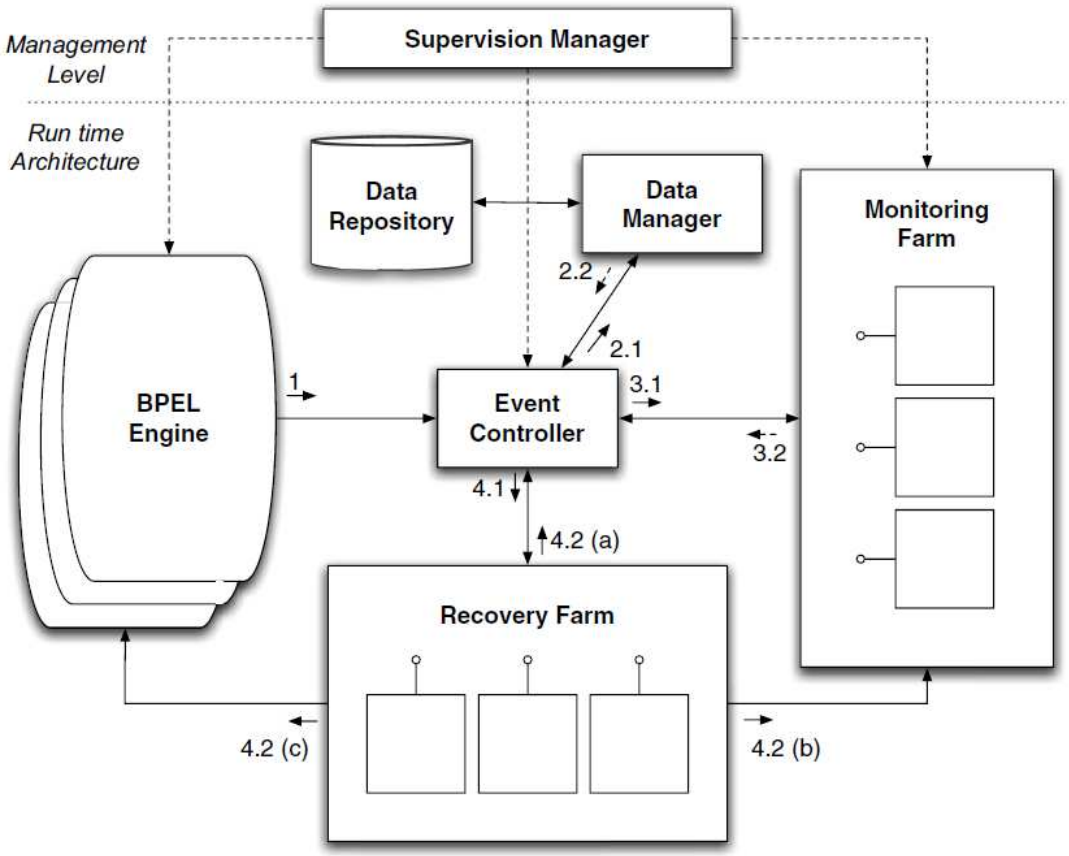

Figura 3.10: Framework unificado.

Antes de iniciar a execução de qualquer processo, o *Supervision Manager* configura os diferentes componentes para que eles executem corretamente as tarefas de supervisão. Após a configuração, toda vez que um processo termina uma atividade, as sondas AOP coletam os dados internos e enviam estes dados para o *Event Controller*, que insere estes dados na sua memória de trabalho. Os dados disponíveis na memória de trabalho do *Event Controller* podem ativar regras adequadas para que o *Data Manager* recupere os dados externos ou históricos e os armazene em sua memória de trabalho.

Depois que os dados são coletados e armazenados, as regras também são responsáveis por enviar os dados para o monitoramento adequado. Isto cria regras especiais que são disparadas pela máquina de regras do *Event Controller*. Para cada tipo de monitoramento utilizado, os projetistas devem especificar asserções que eles querem verificar.

Assim que o resultado do monitoramento é produzido pelo *Monitoring Farm*, ele é enviado para o *Event Controller* que o insere em sua memória de trabalho. Estes novos dados podem disparar regras a pedido do *Recovery Farm* para aplicar algumas ações de recuperação como modificar as instâncias do processo em execução, alterar o funcionamento do *Event Controller*, ou modificar as verificações realizadas pelo monitoramento. Neste ponto, o *Event Controller* tem todos os resultados do monitoramento e pode ativar diferentes estratégias de recuperação, comunicando-se com o *Recovery Farm*. A recuperação no *Recovery Farm* pode acessar a parte interna do processo usando as sondas AOP.

Trabalho de Baresi et al. (2008b)

Um framework de supervisão unificado para monitoramento e recuperação requer um modelo comum para a coleta de dados, uma infra-estrutura comum para gestão, e um framework comum para recuperação. Por outro lado, as técnicas de análise de dados podem ser adicionadas na forma de *plugins*. A estrutura proposta por Baresi et al. (2008b) reflete a separação conceitual entre coleta de dados, análise de dados e recuperação (Figura 3.11).

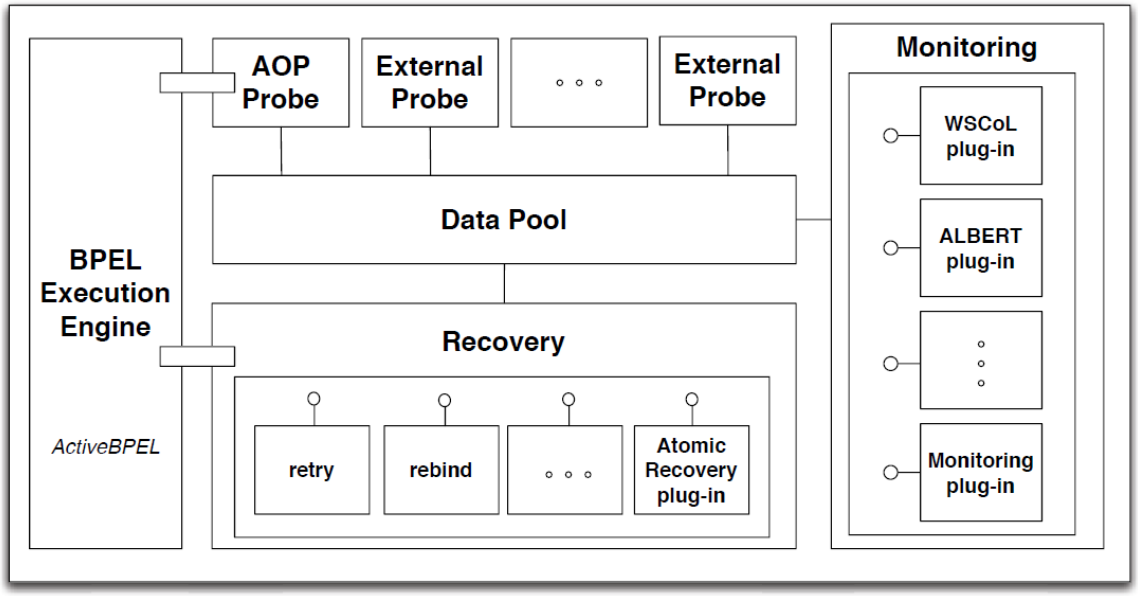

Figura 3.11: Framework unificado.

A coleta de dados explora dois tipos de sondas de dados. O primeiro tipo são sondas AOP. Estas sondas são conectadas diretamente ao ambiente do processo em execução, utilizando técnicas de AOP. Elas param brevemente o processo em execução para recolher o estado interno da execução. A única desvantagem é que para produzir sondas AOP é necessário ter o código fonte da máquina BPEL. O segundo tipo são as sondas externas. Estas sondas podem ser colocadas em todo o ambiente para recolher em tempo de execução as informações do contexto. A vantagem destas sondas é que elas podem ser utilizadas independentemente do ambiente de execução escolhido. Por exemplo, elas podem ser colocadas nos canais de transporte da máquina para obter informações sobre os dados que estão sendo trocados entre parceiros, ou podem ser usadas para detectar eventos da máquina em tempo de execução. A diferença das sondas AOP é que as segundas não podem ser utilizadas para capturar as alterações dos dados internos, como por exemplo, os atributos das atividades BPEL.

Uma vez que os dados são coletados, eles são colocados no *Data Pool*. O *Data Pool* utiliza uma tecnologia que permite separar os dados coletados em duas outras atividades principais: a análise dos dados e o processo de recuperação.

O componente *Monitoring* é configurado para reagir a determinados dados dentro do *Data Pool*. Quando o dado aparece, é passado para a análise. Concluída a análise, os resultados são colocados novamente no *Data Pool* por três razões. A primeira é desbloquear o processo. Isto ocorre quando a natureza da análise é síncrona (por exemplo, pré-condições e pós-condições na chamada de uma atividade BPEL). A segunda é compartilhar os resultados da análise parcial entre os vários tipos de monitoramento, e a terceira é para ativar a recuperação.

Finalmente, o ambiente oferece um sistema uniforme de recuperação, baseado em capacidades atômicas de recuperação que tem acesso à execução do processo usando AOP. Isto permite mudar a forma como o processo irá executar a partir de certo ponto.

Trabalho de Isozaki et al.

Neste trabalho de Isozaki et al. (2008) o monitoramento é realizado no nível de mensagens das atividades já que o envio/recebimento de mensagens ficam registrados. Os processos executando em vários sites com sistemas heterogêneos podem ser observados na Figura 3.12.

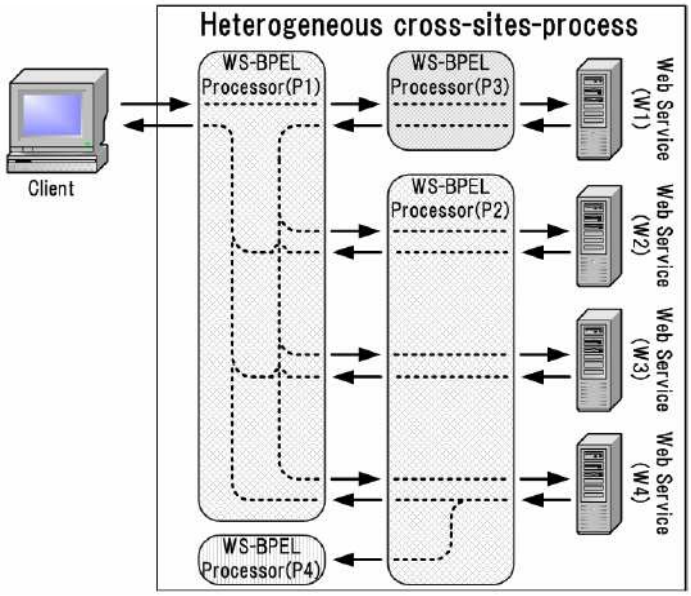

Figura 3.12: Processos heterogêneos entre sites distintos.

Para monitorar os processos heterogêneos executando em múltiplos sites, funções de saída dos dados comuns de auditoria foram implementadas em combinação com todos os processadores. Estas funções são módulos externos que transformam o formato dos dados para o formato especificado pelo fornecedor. Assim os processadores e as descrições dos processos não precisam ser modificados.

VieDAME

No VieDAME (Moser et al., 2008), cada serviço do processo BPEL pode ser marcado como substituível para indicar que serviços alternativos podem ser configurados e invocados ao invés do serviço original definido no processo. Esta arquitetura pode ser observada na Figura 3.13.

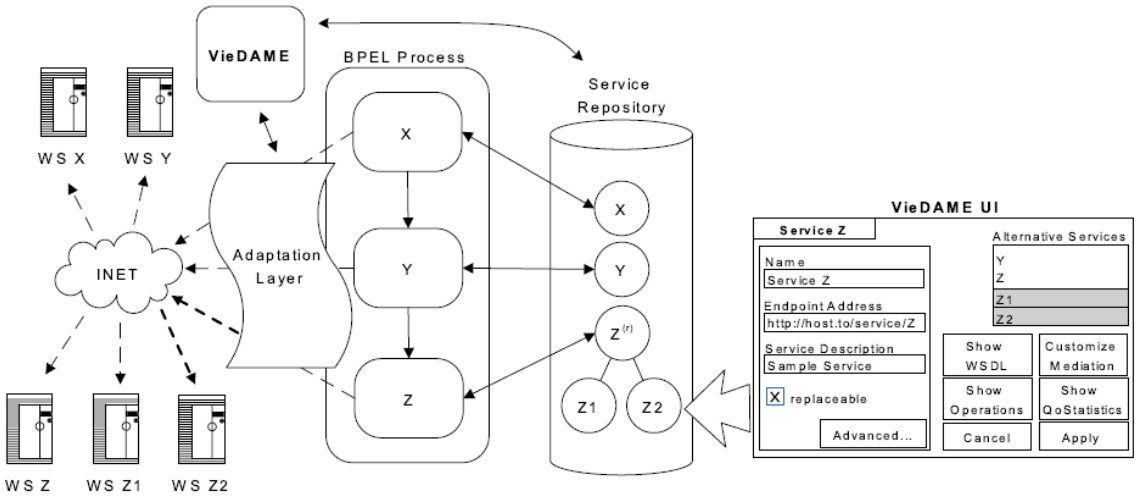

Figura 3.13: Ambiente do VieDAME.

Cada serviço e todos os seus serviços alternativos são armazenados no repositório de serviços do VieDAME. Se um serviço deve ser substituído dinamicamente por um serviço alternativo durante a execução do processo, o serviço original capturado pela camada de adaptação do VieDAME tem de ser marcado como substituível na interface do VieDAME UI. Serviços alternativos podem ser adicionados a qualquer momento. Suas descrições são fornecidas através da VieDAME UI. Além disto, a VieDAME UI pode ser usada para

priorizar os serviços alternativos a serem utilizados, as interfaces dos serviços alternativos não precisam ser iguais a interface do serviço original.

Trabalho de Wu et al.

Nesta proposta de Wu et al. (2008, 2010) os eventos são rastreados durante a execução do processo. O protótipo (ONCE-BPEL) é baseado em aspectos e sua arquitetura é mostrada na Figura 3.14.

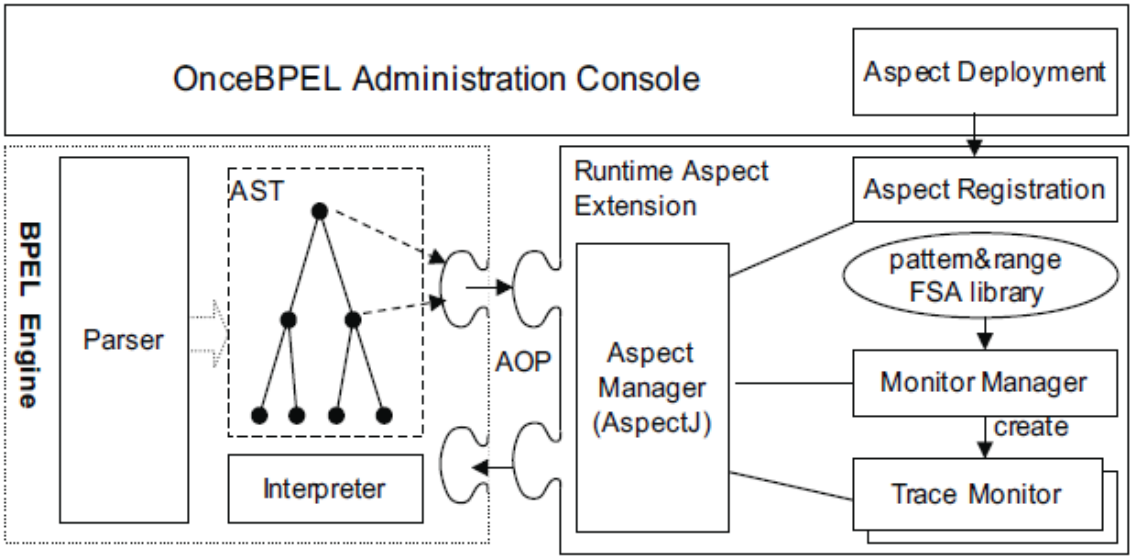

Figura 3.14: Arquitetura do stateful aspect.

Esse sistema oferece aspectos com estado, isto é, o sistema pode observar eventos corridos durante a execução do processo e, quando certos eventos de interesse ocorrem, código extra relacionado com o aspecto é executado. Parâmetros são introduzidos para permitir não só o monitoramento de eventos como de seus valores.

O *Aspect Deployment* especifica como os aspectos deveriam ser aplicados no processo base e isto consiste de duas partes: a instanciação do aspecto e a interação do aspecto. A instanciação é responsável por instanciar um aspecto para um processo concreto. Atualmente, a instanciação do aspecto pode ser a nível do processo e a nível da instância. No nível do processo o aspecto será aplicado a todas as instâncias do processo. No nível da

instância, somente a instância do processo será afetada pelo aspecto. Interação especifica a ordem de execução dos *advices*.

A ferramenta de desenvolvimento do aspecto é uma aplicação web como parte do console de administração do Once-BPEL. Isto permite criar, remover e listar os aspectos atuais.

O *Runtime Aspect Extension* habilita o monitoramento dos eventos a serem rastreados durante a execução do processo e cria os *advices* de forma dinâmica nos processos BPEL. Extensões similares podem ser consideradas para outras implementações da máquina BPEL.

O *Aspect Manager* representa o *advice* principal no ambiente de execução. Este componente tem acesso direto à representação interna do processo e ao estado do processo. Ele é responsável por gerenciar todos os passos do monitoramento.

O *Aspect Registry* salva todas as informações das entidades dos aspectos. Quando o *Stateful Aspect* é implantado, a expressão *pointcut* é avaliada. Esta informação é associada a um *eventcut* e armazenada na estrutura interna. O *Aspect Manager* pode buscar seu conteúdo e a atividade sendo executada.

Antes ou depois que uma atividade é executada, o *aspect manager* é inserido para adiar a execução do aspecto em tempo de execução. Ele procura o símbolo exato do *pointcut* no *aspect registry* com base no contexto de execução atual.

O *aspect manager* mantém um monitor de rastreamento para cada aspecto implantado. Quando um *pointcut* é encontrado, ele irá chamar o monitor de rastreamento para atualizar as instâncias relevantes do monitor. Se não existir um monitor de rastreamento correspondente, o *aspect manager* irá chamar o *monitor manager* para criar um monitor de rastreamento.

Para o *Monitor Manager*, uma importante questão é quando criar um novo monitor de rastreamento. Para resolver este problema, o *monitor manager* salva o evento inicial de cada aspecto implantado. Quando recebe um pedido de criação de um monitor de rastreamento do *aspect manager*, o *monitor manager* irá verificar se o evento é um evento

inicial. Se for, ele cria o novo monitor de rastreamento. Caso contrário, ele descarta o pedido.

Para o *Trace Monitor*, centenas de milhares de instâncias de monitores serão geradas devido ao grande número de parâmetros diferentes durante o tempo de execução, o que sobrecarrega o sistema se a busca não for indexada. Uma busca eficiente por instâncias de monitor é fundamental para diminuir a sobrecarga em tempo de execução. A maior necessidade aqui é localizar rapidamente todas as instâncias relacionadas com um determinado conjunto das instâncias de parâmetros.

Trabalho de Hallé & Villemaire

O objetivo da abordagem de Hallé & Villemaire (2009) são os cenários de serviços web em que a coreografia precisa ser monitorada em tempo de execução. O foco está em um subconjunto da coreografia que contém a sequência das mensagens trocadas de um parceiro específico com seus pares. Esta forma de interação pode ser vista como um "contrato" para ser monitorado e aplicado localmente.

Streaming XML muda a carga de processamento do analisador, que é aliviada da construção da árvore a partir do código XML para a máquina de consulta, que é forçada a atualizar seu estado com base em uma versão linearizada do documento. Como as máquinas de consultas de *streaming* não precisam de todo o documento de uma única vez para ser carregado na memória, elas podem processar o documento em uma ordem maior de magnitude. Uma vantagem crucial do *Streaming XML* é que o resultado da consulta também é em fluxo, ou seja, é possível enviar o resultado de forma progressiva, enquanto a entrada do documento ainda está sendo lida, e sem ter que esperar até o final do documento. Esta característica pode ter um bom uso para o desempenho no monitoramento em tempo de execução.

Um processador de *Streaming XQuery* pode ser usado como um monitor bruto de serviços web em tempo de execução: (i) a partir de uma restrição de monitoramento, constroí uma consulta XQuery *Q* de tal forma que *Q* retorne falso se e somente se a restrição for violada; (ii) computa o resultado de *Q* com uma máquina *Streaming XQuery* no documento formado

de forma progressiva através das mensagens trocadas pelo serviço; e, (iii) gera um erro assim que o falso for lido na saída, parar o monitoramento assim que o verdadeiro for lido na saída (Figura 3.15).

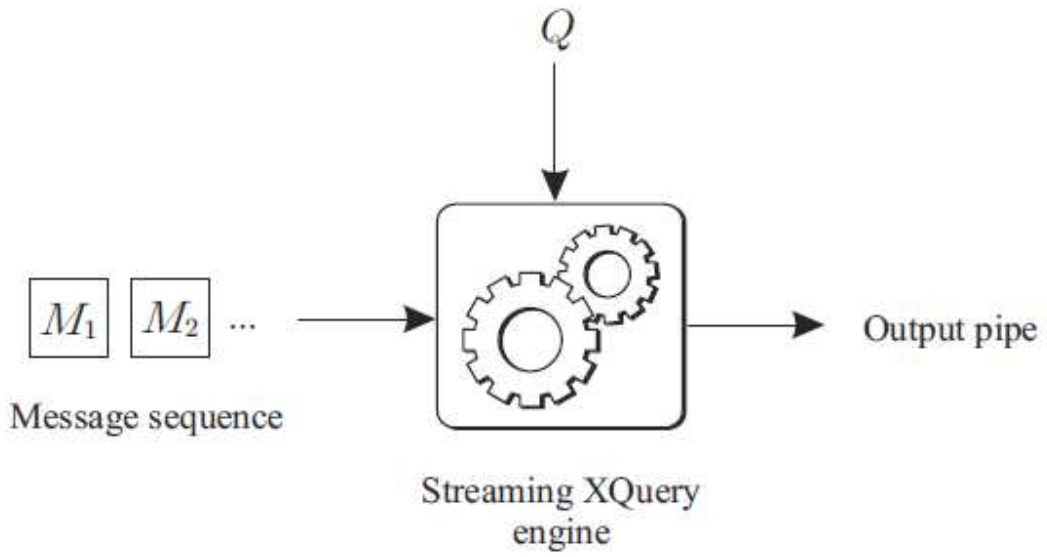

Figura 3.15: Monitoramento em tempo de execução através de streaming XML.

VRESCo

O sistema VRESCo (*Vienna Runtime Environment for Service-oriented Computing*) proposto por Michlmayr et al. (2009) visa combinar as vantagens de ambas abordagens de monitoramento de QoS (*Service-Side* e *Client-Side*). O processamento de eventos é usado para integrar as duas abordagens e fornecer os meios para monitorar os SLAs. Se as obrigações do SLA são violadas, notificações são enviadas para os interessados via email ou através de notificações de serviços web.

Por monitoramento do lado do cliente entende-se por enviar sondas de requisição para o serviço que deveria ser monitorado. Se estas sondas são enviadas, por exemplo, a cada 5 minutos, elas representam somente o estado em cada sonda. Desta forma, o monitoramento do lado do servidor resolve este problema com o monitoramento constante dos atribudos da QoS.

O sistema segue o conceito de "*Software as a Service*" (Software como Serviço) e oferece suas principais funcionalidades como serviços web que podem ser acessados usando bibliotecas de clientes ou diretamente via SOAP. A máquina de consultas oferece uma consulta segura do conteúdo do registro enquanto a camada de controle de acesso é responsável por permitir somente usuários autorizados e autenticados aos serviços do núcleo VRESCo. A máquina de notificação de eventos pode ser usada para informar aos usuários que os eventos de interesse aconteceram (por exemplo, novo serviço é publicado), enquanto todos os eventos são adicionados ao banco de dados de eventos de forma persistente. Esta máquina de eventos é utilizada para o monitoramento do QoS e do SLA. O sistema é apresentado na Figura 3.16.

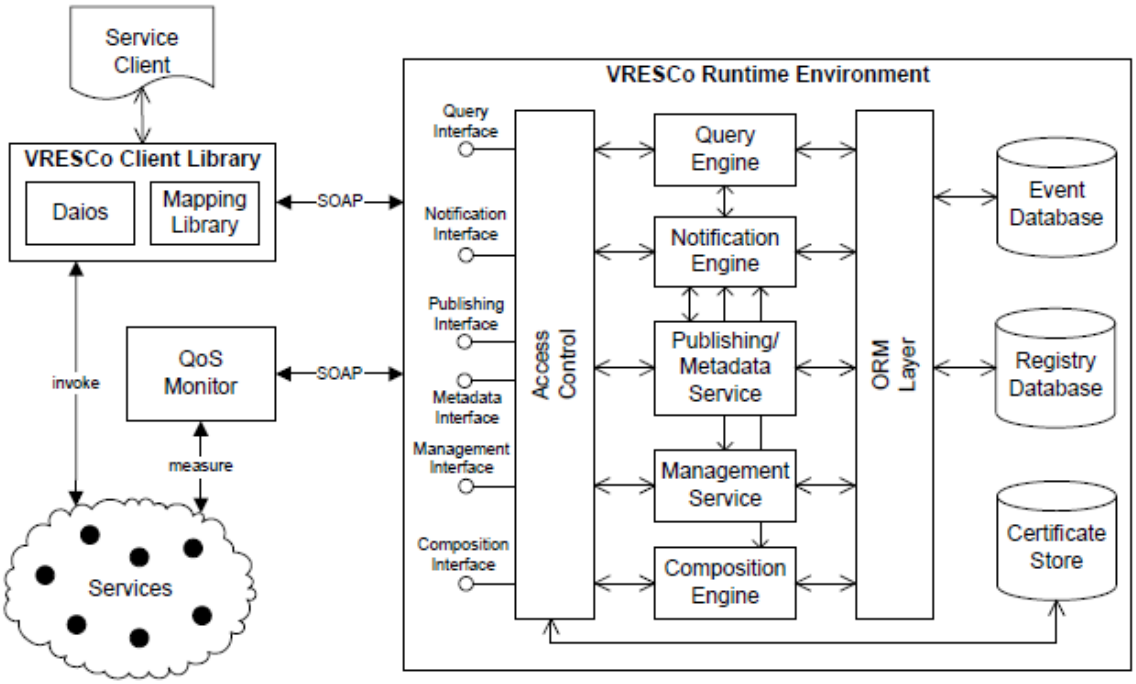

Figura 3.16: Arquitetura do VRESCo.

Os usuários podem especificar o agendamento do monitoramento da QoS definindo qual serviço (ou operação do serviço) deveria ser monitorado e em qual intervalo de tempo. Desde que é uma abordagem do lado do cliente, o monitor executa em uma máquina dedicada. O monitoramento é feito usando AOP e análise de pacotes TCP. Uma vez que os valores da QoS são medidos, eles são publicados dentro do VRESCo. Isto é feito por meio

do gerente da QoS, que por sua vez, publica os eventos correspondentes a QoS dentro da máquina de notificação de eventos. Esta arquitetura pode ser observada na Figura 3.17.

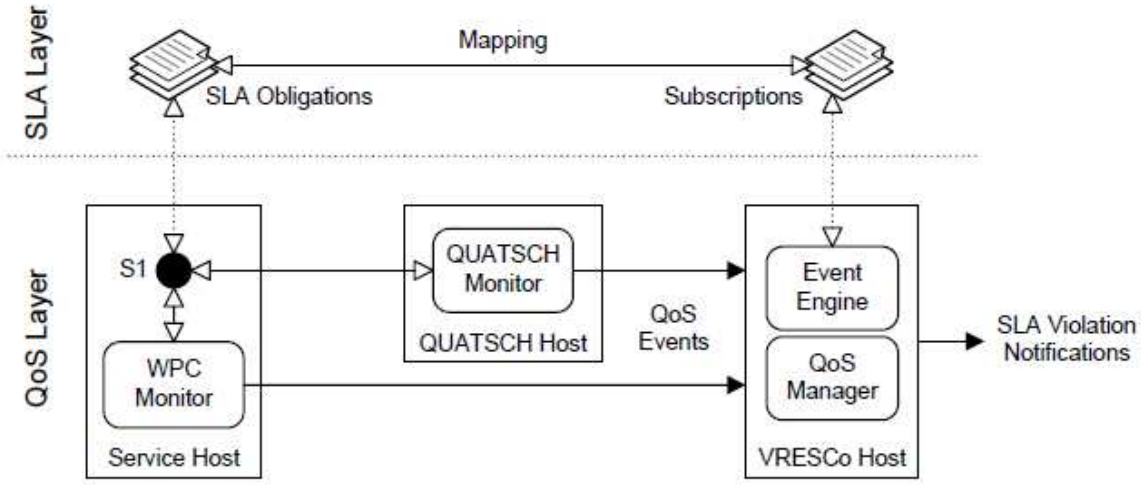

Figura 3.17: Abordagem de monitoramento.

O monitoramento é feito regularmente a partir do agendamento de monitoramento definido pelo usuário. O conjunto dos resultados dos eventos de QoS representam a história da informação da QoS como uma coleção de vários instantes independentes da QoS. Para tornar estes valores facilmente acessíveis, eles são agregados por uma tarefa de agendamento agregador da QoS e inserido no serviço (ou operação do serviço) correspondente.

Trabalho de Wetzstein et al.

O framework de Wetzstein et al. (2009) pode ser dividido em três camadas diferentes, como representado na Figura 3.18. As camadas são descritas a seguir.

Process Runtime Layer. Nesta camada, os processos de negócio WS-BPEL são definidos e executados. O processo pode ser executado na máquina padrão WS-BPEL compatível, desde que a máquina seja capaz de fornecer as informações do processo que são necessárias para o cálculo dos PPMs (*Process Performance Metrics*) na forma dos eventos dos processos.

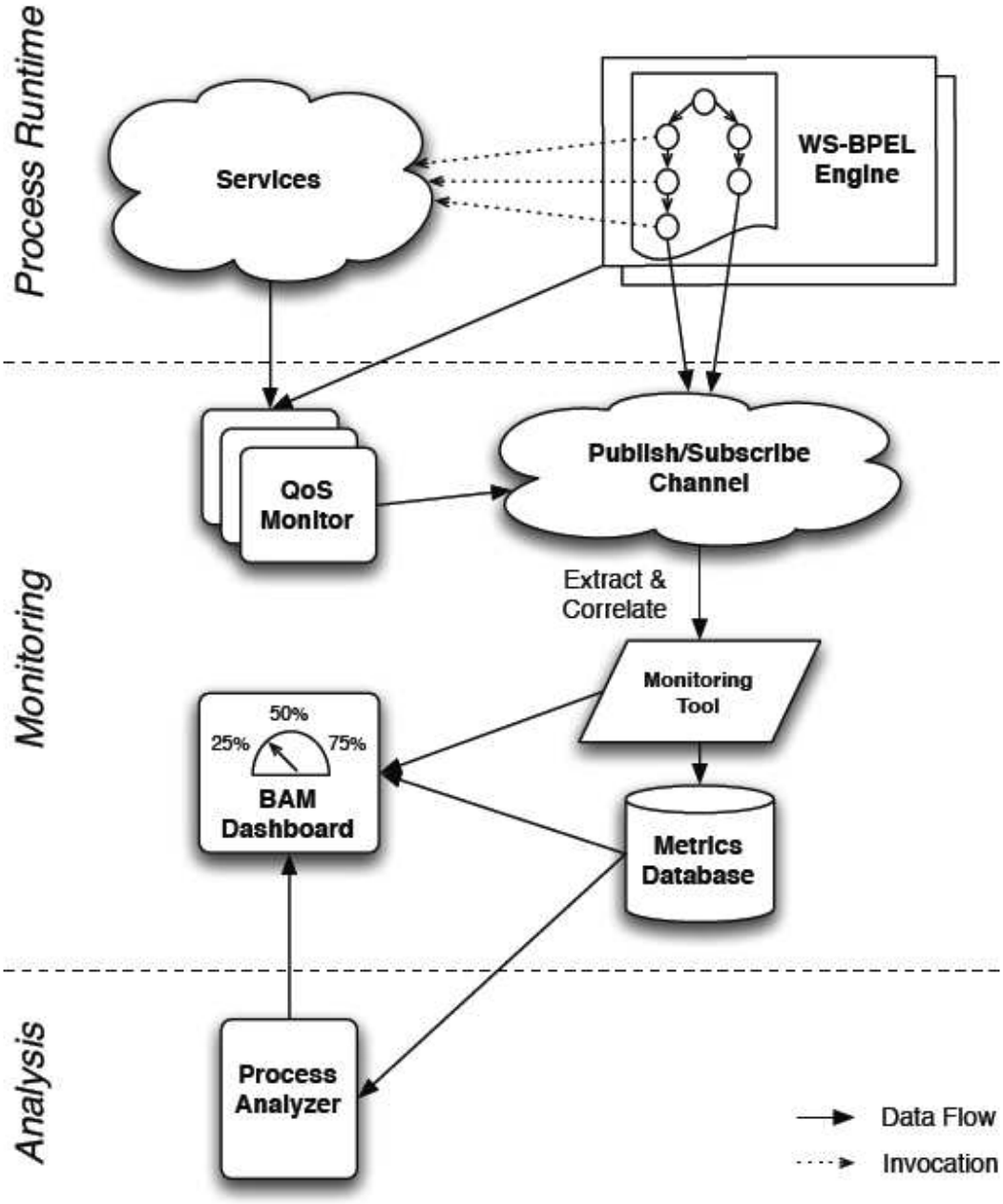

Figura 3.18: Framework de monitoramento e análise.

Monitoring Layer. Nesta camada, as informações sobre os processos de negócio em execução e sobre os serviços que interagem são coletadas para monitorar os KPIs (*Key Performance Indicators*), os PPMs e as métricas de QoS. Durante o tempo de execução do

processo, o monitor QoS e a máquina do processo WS-BPEL publicam os eventos para um canal de publicação/registro em que a ferramenta de monitoramente está inscrita. O PPM e as métricas do QoS são calculados, armazenados no banco de dados de métricas para análise posterior e disponibilizados no painel do BAM (*Business Activity Monitoring*).

Process Analysis Layer. Nesta camada, a informação das métricas coletadas é armazenada pelo componente analisador de processo. Quando o usuário está interessado em fazer uma análise de dependência dos KPIs, por exemplo, o analisador de processo reúne os dados necessários da métrica no banco de dados de métricas, prepara os dados, e usa um algoritmo de árvore de decisão para gerar a árvore de dependência que mostra os fatores influentes do KPI. Os resultados da análise são novamente mostrados no painel para os usuários do sistema, que podem usar esta informação para melhorar o processo de negócio.

SECMOL

O framework SECMOL (Baresi et al., 2010) é organizado em três elementos chave. *Data Collectors* que coletam e extraem os dados necessários para fazer o monitoramento, *Data Analyzers* que analisam estes dados e *Monitor Manager* que integra e supervisiona todo o processo de monitoramento (Figura 3.19).

O *Monitoring Manager* é fixo e obrigatório, ao passo que é possível integrar vários *collectors* e *analyzers* diferentes. SECMOL fornece uma sintaxe bem definida baseada em XML para processar todos os aspectos relacionados com o monitoramento.

O *Monitoring Manager* supervisiona e coordena todo o processo de monitoramento. Ele recebe as especificações SECMOL, avalia se elas podem ser monitoradas, e identifica os coletores e analisadores de dados para adquirir informações e verificar as condições como restrições funcionais e não funcionais. O suporte para o monitoramento interno vem das sondas AOP para o processo BPEL que será monitorado. Estas sondas coletam os dados durante a execução e os enviam para os componentes de análise apropriados. Por outro lado, o suporte para o monitoramento externo vem dos coletores de dados que são externos e executam em paralelo com o processo.

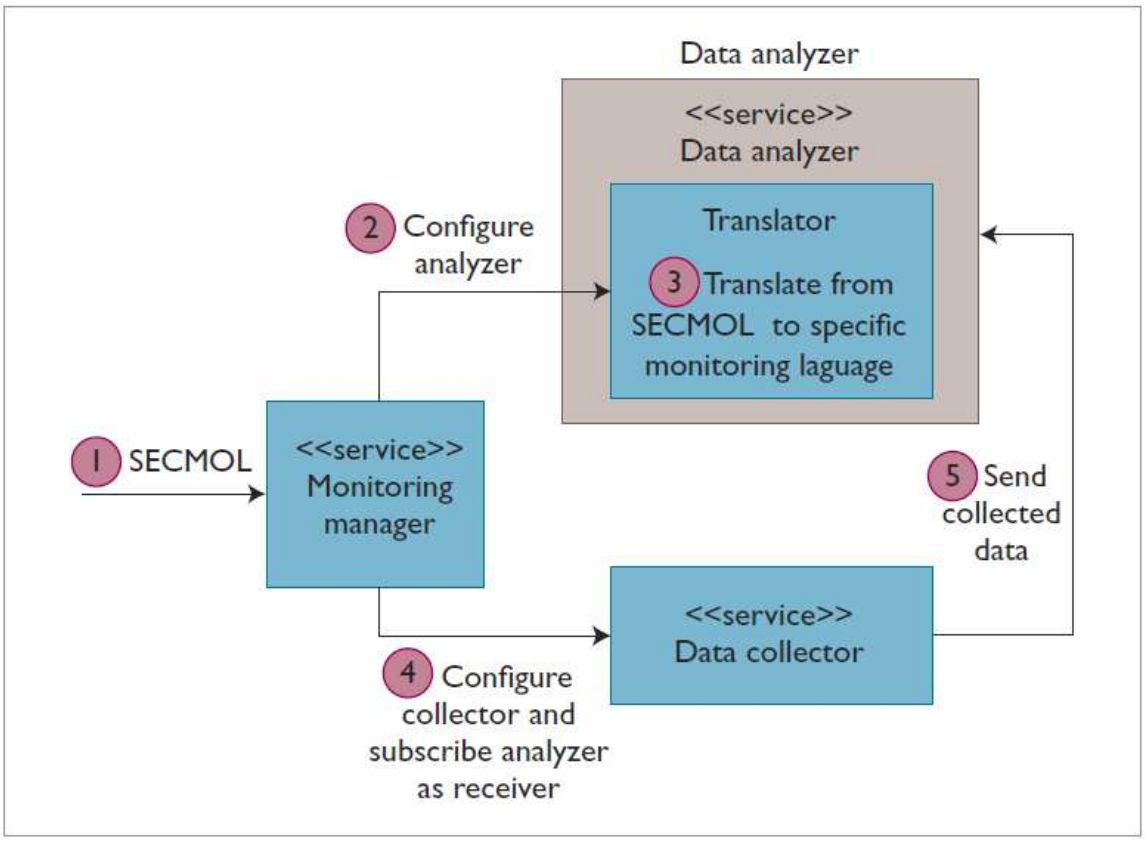

Figura 3.19: Arquitetura do framework de monitoramento.

Depois de preparar toda a infra-estrutura, o *Monitoring Manager* também é responsável por comunicar os resultados aos interessados.

Uma especificação de monitoramento em SECMOL consiste de duas partes:

- *Monitoring Policies* que especificam como implementar os componentes de monitoramento, quando as regras de monitoramento devem ser verificadas, como elas devem ser monitoradas, e como os resultados do processo de monitoramento devem ser apresentados.
- *Monitoring Rules* que expressam as condições que o sistema deve satisfazer. Estas condições podem expressar tanto restrições funcionais como propriedades de QoS.

Trabalho de Bratanis et al.

O foco do trabalho de Bratanis et al. (2010) é na integração de abordagens de monitoramento diferentes. A arquitetura (Figura 3.20) é modular e independente do fornecedor, de modo que diferentes fornecedores são suportados.

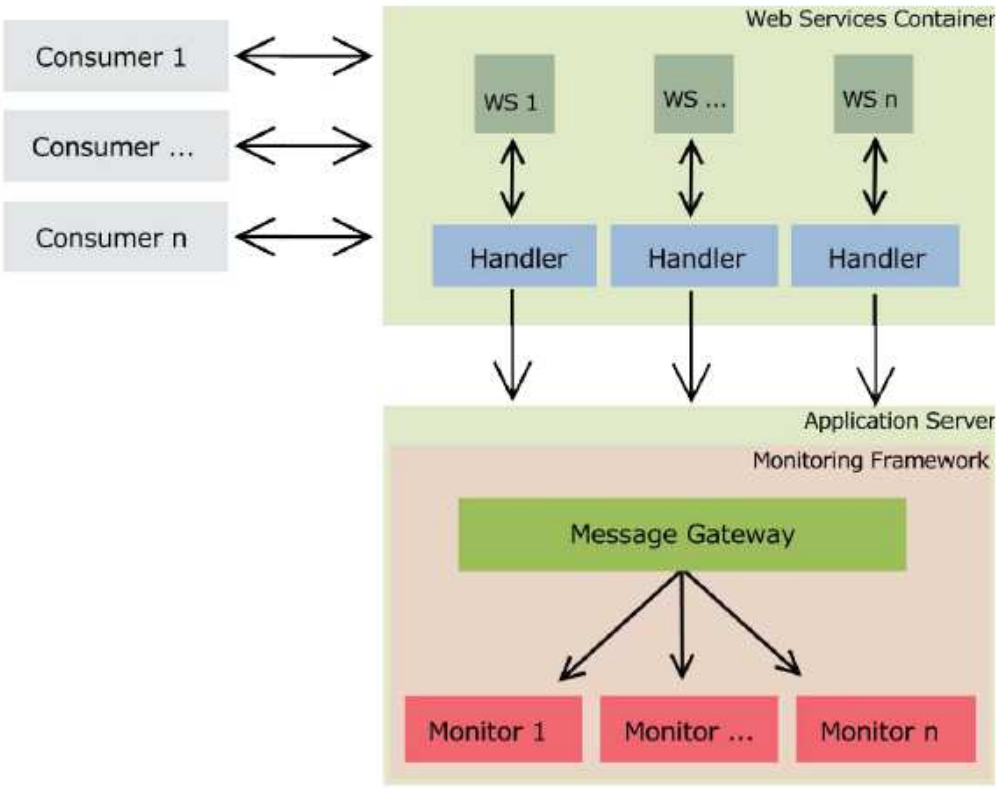

Figura 3.20: Componentes da arquitetura de monitoramento.

A arquitetura de monitoramento tem somente um serviço para interceptar as mensagens. A interceptação das mensagens de requisição/resposta, juntamente com o gerenciamento da sessão, registro das ações e outras funcionalidades foram incorporadas na plataforma, que fornece vários monitores de baixa complexidade já que utilizam os recursos da plataforma e tratam aspectos específicos.

Os *Handlers* são usados para interceptar o tráfego dos serviços web. Um ou mais *Handlers* podem ser associados a cada serviço web. O *Handler* envia as mensagens interceptadas para o framework de monitoramento, implantado no servidor de aplicação, por meio do

Message Gateway (MG). O MG é um serviço web que recebe e envia as mensagens, interceptadas pelos *handlers*, para cada monitor individual, que é responsável por diferentes aspectos do monitoramento.

Trabalho de Lomuscio et al.

A abordagem de Lomuscio et al. (2010) é para o monitoramento do comportamento local do processo. Para cada agente a ser monitorado, todos os possíveis comportamentos são representados como um TADD (*timed automata with discrete data*) e armazenados no verificador. O mecanismo de monitoramento verifica os registros contra as especificações TADD e relata se o comportamento em tempo de execução está de acordo como previsto contratualmente. Esta arquitetura pode ser observada na Figura 3.21.

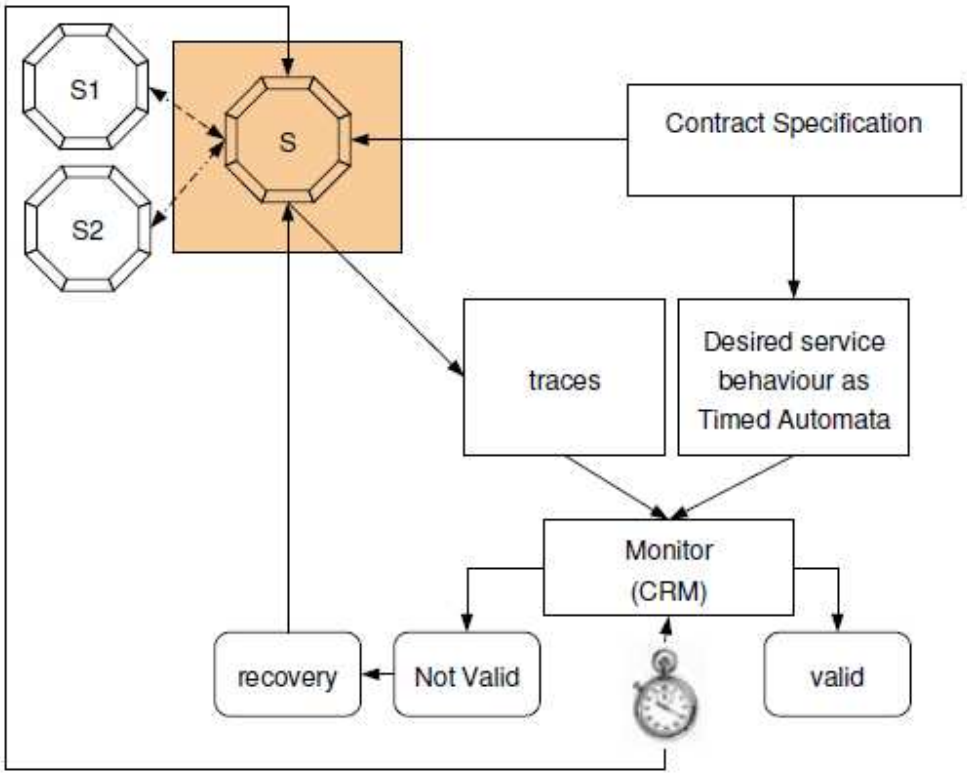

Figura 3.21: Arquitetura e metodologia da proposta.

É utilizado o formato XML para o verificador representado em TADD. A especificação TADD contém todas as possibilidades desejáveis de comportamento do serviço. Normalmente, um conjunto completo de comportamentos para os serviços podem ser derivados de: (i) Seus comportamentos contratuais compatíveis. Estes comportamentos encapsulam obrigações contratuais para o serviço; (ii) Comportamentos classificados como violação do contrato; e, (iii) Comportamentos que definem uma recuperação a partir das violações ocorridas.

A máquina de monitoramento é o componente principal responsável por testar a conformidade do comportamento do serviço previsto na especificação TADD do serviço.

Resultados dos relatórios em tempo de execução podem ser analisados de várias formas. No caso de transições no TADD de acordo com o contrato, o serviço pode continuar a execução na orquestração. Para transições com violação de contrato, o administrador do serviço pode impor sobre o serviço uma tarefa de recuperação prevista. Nos outros casos o administrador pode escolher em substituir as violações relatadas e permitir que o serviço continue a execução. Para transições com contínua violação de contrato, o serviço pode ser parado. E os registros podem ser armazenados em arquivos para futura análise.

4. Comparativo

Nesta seção é apresentada uma tabela comparativa entre as abordagens de monitoramento em serviços web com diferentes formas de monitoramento.

Legenda da Tabela:

X	Característica apresentada na abordagem de forma nítida.
?	Característica deduzida da abordagem pelos autores deste estudo.
[01]	Tian et al., 2004.
[02]	Baresi & Guinea, 2005.
[03]	Baresi & Guinea, 2007; Baresi et al., 2007.
[04]	Baresi & Guinea, 2008.
[05]	Beeri et al., 2007a, 2007b, 2008.
[06]	Bianculli & Ghezzi, 2007.
[07]	Baresi et al., 2008a.
[08]	Baresi et al., 2008b.
[09]	Isozaki et al., 2008.
[10]	Moser et al., 2008.
[11]	Wu et al., 2008, 2010.
[12]	Hallé & Villemaire, 2009.
[13]	Michlmayr et al., 2009.
[14]	Wetzstein et al., 2009.
[15]	Baresi et al., 2010.
[16]	Bratanis et al., 2010.
[17]	Lomuscio et al., 2010.

Tabela 1: Comparativo entre as abordagens de monitoramento e as formas de monitoramento.

Abordagem	Intrusão			Carga		Nível				Interceptação			Tempo	
	Intrusivo	Levemente Intrusivo	Não-Intrusivo	Pesado	Leve	Site	Processo	Mensagem	Atividade	Handler	Wrapper	Proxy	Síncrono	Assíncrono
[01]			?	?		?					?		X	
[02]		X		X			X					X	X	
[03]		X		X			X					X	X	
[04]	X			X					X	X				X
[05]	X			X			X			X			X	
[06]			X	?			X				X		X	
[07]			X	X		X	X		X		X		X	X
[08]			X	X	X		X				X		X	X
[09]		X			X			X				X	X	
[10]		X		X		X	X					?	X	
[11]			?		X		X		X		?		X	
[12]		?		X		X	X					?	X	
[13]		X		X			X					?	X	
[14]			?	X			X				?		X	X
[15]			?	X		X	X	X			?		X	
[16]	?				X			?		X			X	
[17]		?		X			X					?	X	

Referências

Alonso G., Casati F., Kuno H., Machiraju V. 2010. *Web Services: Concepts, Architectures and Applications* (1st ed.). Springer Publishing Company, Incorporated.

Angelov S. & Grefen P. 2008. An e-contracting reference architecture. *J. Syst. Softw.* 81, 11 (November 2008), 1816-1844.

Baresi L., Ghezzi C., Guinea S. 2004. Smart monitors for composed services. In *Proceedings of the 2nd international conference on Service oriented computing* (ICSOC '04). ACM, New York, NY, USA, 193-202.

Baresi L. & Guinea S. 2005. Towards dynamic monitoring of WS-BPEL processes. In *Service Oriented Computing - ICSOC 2005, Third International Conference*, Amsterdam, The Netherlands, December 12-15, 2005, volume 3826 of Lecture Notes in Computer Science, pages 269–282. Springer.

Baresi L. & Guinea S. 2007. Dynamo and Self-Healing BPEL Compositions. In Companion to the proceedings of the 29th International Conference on Software Engineering (ICSE COMPANION '07). IEEE Computer Society, Washington, DC, USA, 69-70.

Baresi L. & Guinea S. 2008. A dynamic and reactive approach to the supervision of BPEL processes. In Proceedings of the 1st India software engineering conference (ISEC '08). ACM, New York, NY, USA, 39-48.

Baresi L., Guinea S., Nano O., Spanoudakis G. 2010. Comprehensive Monitoring of BPEL Processes. *IEEE Internet Computing* 14, 3 (May 2010), 50-57.

Baresi L., Guinea S., Pasquale L. 2008a. Integrated and Composable Supervision of BPEL Processes. In *Proceedings of the 6th International Conference on Service-Oriented Computing* (ICSOC '08), Athman Bouguettaya, Ingolf Krueger, and Tiziana Margaria (Eds.). Springer-Verlag, Berlin, Heidelberg, 614-619.

Baresi L., Guinea S., Pasquale L. 2008b. Towards a unified framework for the monitoring and recovery of BPEL processes. In *Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications* (TAV-WEB '08), Tevfik Bultan and Tao Xie (Eds.). ACM, New York, NY, USA, 15-19.

Baresi L., Guinea S., Plebani M. 2007. Business process monitoring for dependability. In *Architecting dependable systems IV*, Rogério de Lemos, Cristina Gacek, and Alexander Romanovsky (Eds.). Lecture Notes In Computer Science, Vol. 4615. Springer-Verlag, Berlin, Heidelberg 337-361.

Beeri C., Eyal A., Milo T., Pilberg A. 2007a. Monitoring business processes with queries. In Proceedings of the 33rd international conference on Very large data bases (VLDB '07). VLDB Endowment 603-614.

Beeri C., Eyal A., Milo T., Pilberg A. 2007b. Query-based monitoring of BPEL business processes. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD '07). ACM, New York, NY, USA, 1122-1124.

Beeri C., Eyal A., Milo T., Pilberg A. 2008. BP-Mon: query-based monitoring of BPEL business processes. SIGMOD Rec. 37, 1 (March 2008), 21-24.

Bianculli D. & Ghezzi C. 2007. Monitoring conversational web services. In *2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting* (IW-SOSWE '07). ACM, New York, NY, USA, 15-21.

Bratanis K., Dranidis D., Simons A. 2010. An extensible architecture for run-time monitoring of conversational web services. In *Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond* (MONA '10). ACM, New York, NY, USA, 9-16.

Chinnici R., Moreau J., Ryman A., Weerawarana S. 2006. Web Services Description Language (WSDL) version 2.0 part 1: Core language. http://www-mit.w3.org/TR/2006/CR-wsdl20-20060327/wsdl20-z.pdf Acesso em 2011-09-01.

Hallé S. & Villemaire R. 2009. Runtime monitoring of web service choreographies using streaming XML. In *Proceedings of the 2009 ACM symposium on Applied Computing* (SAC '09). ACM, New York, NY, USA, 2118-2125.

Isozaki Y., Kanna Y., Kato K., Kanai T., Miyamoto D, Kikuchi S. 2008. Monitoring Cross-Site Processes Executed across Heterogeneous WS-BPEL Processors. In Proceedings of the 2008 12th Enterprise Distributed Object Computing Conference Workshops (EDOCW '08). IEEE Computer Society, Washington, DC, USA, 389-392.

Keller A. & Ludwig H. 2003. "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services", Journal of Network and Systems Management, 11(1), Springer, pp. 57-81.

Kiczales G. 1996. Aspect-oriented programming. *ACM Comput. Surv.* 28, 4es, Article 154 (December 1996).

Lomuscio A., Solanki M., Penczek W., Szreter M. 2010. Runtime monitoring of contract regulated web services. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1 (AAMAS '10), Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1449-1450.

Michlmayr A., Rosenberg F., Leitner P., Dustdar S. 2009. Comprehensive QoS monitoring of Web services and event-based SLA violation detection. In Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing (MWSOC '09). ACM, New York, NY, USA, 1-6.

Moser O., Rosenberg F., Dustdar S. 2008. VieDAME - flexible and robust BPEL processes through monitoring and adaptation. In Companion of the 30th international conference on Software engineering (ICSE Companion '08). ACM, New York, NY, USA, 917-918.

OASIS. 2004. Uddi version 3.0.2. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm Acesso em 2011-09-01.

OASIS. 2007. Business process execution language for web services version 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html Acesso em 2011-09-01.

Papazoglou M. 2007. Web Services: Principles and Technology. Prentice Hall, 1 edition, September.

Sahai A., Machiraju V., Sayal M, Moorsel A. P. A. V., Casati F. 2002. Automated SLA Monitoring for Web Services. In *Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications* (DSOM '02), Metin Feridun, Peter G. Kropf, and Gilbert Babin (Eds.). Springer-Verlag, London, UK, 28-41.

Santos L. L. 2011. Monitoramento de contratos eletrônicos baseados em características. *Dissetação de Mestrado*. Universidade Estadual de Campinas (UNICAMP), Instituto de Computação (IC), Campinas, SP.

Tian M., Gramm A., Ritter H., Schiller J. 2004. Efficient Selection and Monitoring of QoS-Aware Web Services with the WS-QoS Framework. In *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence* (WI '04). IEEE Computer Society, Washington, DC, USA, 152-158.

W3C. 2007. SOAP Wersion 1.2 part 1: Messaging framework (second edition). http://www.w3.org/TR/soap12-part1/ Acesso em 2011-09-01.

W3C. 2008. eXtensible Markup Language (XML) 1.0 (fifth edition). http://www.w3.org/TR/1999/REC-xslt-19991116 Acesso em 2011-09-01.

Weske M. 2007. Business Process Management: Concepts, Languages, Architectures. Springer Verlag, first edition, November.

Wetzstein B., Leitner P., Rosenberg F., Brandic I., Dustdar S., Leymann F. 2009. Monitoring and analyzing influential factors of business process performance. In Proceedings of the 13th IEEE international conference on Enterprise Distributed Object Computing (EDOC'09). IEEE Press, Piscataway, NJ, USA, 118-127.

Wu G., Wei J., Huang T. 2008. Flexible Pattern Monitoring for WS-BPEL through Stateful Aspect Extension. In Proceedings of the 2008 IEEE International Conference on Web Services (ICWS '08). IEEE Computer Society, Washington, DC, USA, 577-584.

Wu G., Wei J., Ye C., Zhong H., Huang T. 2010. Detecting Data Inconsistency Failure of Composite Web Services Through Parametric Stateful Aspect. In Proceedings of the 2010 IEEE International Conference on Web Services (ICWS '10). IEEE Computer Society, Washington, DC, USA, 68-75.