Interação de características na composição de serviços web: prevenindo a violação dos requisitos do usuário

Roberto C. Figueiredo[1], Daniela B. Claro[2], Raimundo A. Macêdo[3], Aline S. Andrade[4]

[1]Departamento de ciência da computação – Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n, Ondina – Salvador – BA – Brazil

{roberedo,dclaro, macedo, aline}@ufba.br

Abstract. *Feature interaction is an undesirable interaction between services of a composition which may violate the functional and non-functional user's requirements. Due to the dynamic nature, heterogeneity and openness of web services, solve feature interaction is a complex task because it is difficult to control services that was developed by differents vendors. There is no access to such web service implementations. The great challenge is feature interaction prevention. It can be done in online or offline mode, however, there aren't works that prevent in online mode. In this article, an autonomic mechanism, based on neural networks and genetic algorithms was proposed to prevent feature interaction in web services composition. The results demonstrates a reaction time and accuracy appropriate to monitor and detect the causes of feature interaction causes in order to facilitate the prevention.*

Resumo. *Interação de característica é uma interação indesejável entre os serviços de uma composição, que pode violar os requisitos funcionais e não funcionais do usuário. Devido à natureza dinâmica, heterogeneidade e abertura da composição de serviços web, solucionar interação de característica é uma tarefa complexa, pois é difícil controlar os serviços que pertencem a diferentes domínios administrativos - não há acesso às suas implementações. O grande desafio é a prevenção da ocorrência de interação de característica. Isto pode ser feito pelo uso de técnicas online ou offline, no entanto, não foram encontrados trabalhos que previnam de modo online. Neste artigo, Um mecanismo autonômico e online, baseado em Redes Neurais e Algoritmos Genéticos, foi proposto, para prevenir interações de características na composição de serviços web. Os resultados demonstram um tempo de reação e precisão adequados para monitorar e detectar as causas de interações de características no intuito de viabilizar a prevenção.*

1. Introdução

Um serviço web é uma funcionalidade de um sistema de informação publicada na internet [Alonso 2004]. Usuários buscam serviços web na internet de modo que os seus requisitos funcionais e não funcionais sejam alcançados [Weiss et al. 2007]. Frequentemente, alcançar requisitos funcionais e não funcionais do usuário requer a combinação de diferentes serviços web, denominada composição [Amorim et al. 2011].

Uma composição de serviços web pode ser constituída por serviços de alto nível e serviços de baixo nível. Serviços de alto nível, também denominados de serviços abstratos, representam os requisitos funcionais e/ou não funcionais do usuário. Estes serviços

utilizam os de baixo nível, também denominados serviços concretos, para implementar a funcionalidade desejada, de modo que os requisitos do usuário sejam alcançados. Serviços concretos podem participar de composições implícitas ou não planejadas e ocasionar efeitos colaterais que podem comprometer os requisitos funcionais e não funcionais do usuário [Weiss e Esfandiari 2004].

Efeitos colaterais podem ser definidos como efeitos indesejáveis originados de interações indesejáveis entre serviços concretos que participam de uma composição implícita ou não planejada. Na literatura, interações indesejáveis são denominadas interação de características [Weiss et al. 2007].

Para avaliar o problema de interações de características, os desafios de prevenir, detectar e resolver são os principais direcionamentos de pesquisa. A prevenção busca um caminho no projeto da composição para evitar a ocorrência de interações de características. A detecção assume que a interação de características existe e define métodos para identifica-la e localiza-la. Por fim, resolver interações de características significa minimizar ou eliminar os seus efeitos indesejáveis após a ocorrência de uma detecção.

Devido à dinamicidade, heterogeneidade e a abertura da web, prevenir interações de características é uma tarefa complexa. Nesse contexto, pouco se fez para prevenir interações de características na composição de serviços abstratos. A prevenção pode ser implementada no projeto da composição ou na definição de uma arquitetura que controle as interações entre serviços. A prevenção offline (em tempo de projeto) torna-se uma solução incompleta, pois a dinâmica da web faz surgir novos casos de interações de características. A prevenção online é uma solução desejável e que pode viabilizar o alcance dos requisitos do usuário. Para prevenir é preciso identificar as causas para a ocorrência de interação de característica. As principais causas para a ocorrência de interações de características são a contenção de recursos, violação de hipóteses e conflito entre metas [Cameron e Velthuijsen 1993].

Assim, o presente trabalho propõe um método de prevenção ou minimização de interações de características online na composição abstrata do usuário. Isto pode possibilitar uma melhora na qualidade de serviço e diminuição da frequencia de violações dos requisitos funcionais e não funcionais do usuário. Consequentemente, serviços concretos pertencentes a diferentes domínios podem ter minimizada a necessidade de resolver interações de características indesejáveis, pois a frequencia de detecções pode ser reduzida.

O presente artigo está organizado em seções das quais a a seção 2 trata da definição de interação de características. A seção 3 descreve exemplos relacionados aos efeitos colaterais que podem ocorrer entre serviços concretos. Trabalhos relacionados aos principais desafios para solucionar interações de características são discutidos na seção 4. Gestores autonômicos são apresentados como proposta na seção 5. Em seguida, os dados da pesquisa são discutidos na seção 6. Enfim, uma conclusão é elaborada.

2. Interação de Característica

Segundo autores em [Crespo et al. 2007], uma característica é uma unidade de funcionalidade de um sistema. Autores em [Weiss et al. 2007] definem uma característica como a menor unidade de serviço visível aos usuários. Usuários utilizam as características de modo que os seus requisitos sejam alcançados. No entanto, quando ocorre

uma composição de características, interações indesejáveis, denominadas de interações de características, podem ocorrer. Assim, segundo [Calder et al. 2003], dado duas características S_1 e S_2 e se cada característica satisfizer as suas propriedades ou requisitos T_1 e T_2, respectivamente, ou seja, $(S_1 \models T_1) \wedge (S_{2,} \models T_2)$, então uma interação de características ocorre quando a composição dessas características não satisfizerem um ou mais requisitos, ou seja, $S_1 \oplus S_2 \not\models T_1 \wedge T_2$. Isto significa que após a composição de S_1 e S_2 pelo menos uma das propriedades T_i desta formula será falsa para $i = 1, 2$.

Diversos motivos podem ocasionar o surgimento de interação de características na composição web. Autores em [Cameron e Velthuijsen 1993] definem as principais causas como a contenção de recursos, violação de hipóteses e conflito entre metas:

- Contenção de recursos: A contenção de recursos é caracterizada pela competição entre serviços por recursos limitados. Pode ocorrer a violação do requisito do serviço que não obtiver o recurso.
- Violação de hipóteses: Os projetistas de serviços fazem hipóteses sobre a interação do serviço com outros na composição, no entanto, tais hipóteses podem ser violadas se novos serviços surgirem na composição.
- Conflito entre metas: Serviços com metas incompatíveis. Normalmente, requisitos não funcionais incompatíveis proporcionam conflito entre metas. (e.g, segurança e desempenho são conflitantes)

2.1. Interação de características na composição Web

A etapa inicial para o projeto de uma composição web consiste na definição de um fluxo de serviços $T1 \rightarrow T2 \rightarrow T_n$ denominado de composição de serviços abstratos (requisitos). Para cada serviço abstrato, a meta é encontrar serviços concretos (implementação do serviço) $S1, S2, \ldots, S_n$ que proporcionem o alcance dos requisitos desejados. No entanto, serviços concretos podem pertencer a diferentes domínios e então interagir com outros serviços concretos de modo não planejado. Essas interações são indesejáveis, pois efeitos colaterais podem violar os requisitos da composição de serviços abstratos.

Assim, efeitos colaterais podem ser considerados como um efeito negativo originado de uma composição de serviços não planejada [Weiss e Esfandiari 2004]. As composições podem ser explícitas e planejadas ou implícitas e não planejadas. Segundo [Weiss e Esfandiari 2004], efeitos colaterais ocorrem em composições implicitas e não planejadas de serviços concretos e impactam os requisitos dos serviços abstratos que utilizam tais serviços concretos.

3. Exemplos de Interação de Característica na Composição de Serviços Web

Nesta seção, serão ilustrados exemplos de interações de características ou efeitos colaterais na composição web, sendo estes relacionados às causas de contenção de recursos, violação de hipóteses e conflito entre metas.

3.1. Contenção de recursos

A Figura 1 apresenta um exemplo de violação de requisitos não funcionais baseados em métricas de qualidade de serviço (QoS). As métricas de tempo de resposta e custo são desejáveis em diferentes composições. Os requisitos para as composições são estabelecidos e as composições são executadas.

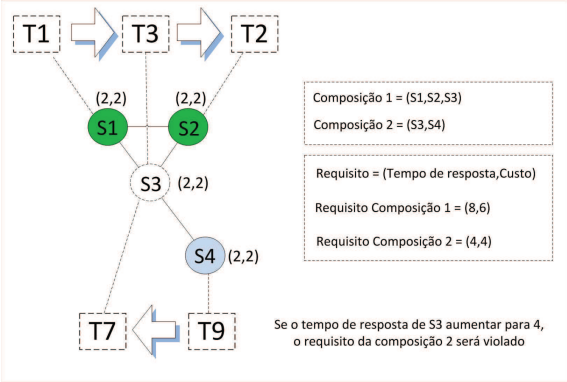

Figura 1. Contenção de recursos na Composição de Serviços Web

Uma degradação de algum serviço da composição pode ocasionar a violação do requisito global da composição. Nesse exemplo, adaptado de autores em [Nguyen e Kowalczyk 2006], uma composição abstrata de serviços $T1 \rightarrow T3 \rightarrow T2$ seleciona os serviços $S1$,$S3$ e $S2$, respectivamente para cada funcionalidade T desejada. Outra composição abstrata $T9 \rightarrow T7$ seleciona os serviços $S4$ e $S3$. Devido ao fato de que ambas as composições competem pelo serviço $S3$, este pode ter o seu tempo de resposta degradado devido a uma contenção de recursos. Ou seja, as duas composições interagem através do serviço $S3$, sobrecarregando-o. O requisito da composição $T9 \rightarrow T7$ é então violado, pois o tempo de resposta desejado não será cumprido.

Neste exemplo, uma composição implícita pode ser caracterizada pela interação entre as composições e o serviço $S3$. Dada a a composição $C1 = T1 \rightarrow T3 \rightarrow T2$ e a composiçao $C2 = T9 \rightarrow T7$ então tem-se uma composição implícita $C1 \rightarrow C2$. Esta composição ocasiona um efeito colateral em $T7$, devido ao aumento do tempo de resposta de $S3$ para 4 segundos. Este efeito faz com que o requisito $T7$ seja violado e consequentemente, o requisito global de $C2$.

3.2. Violação de hipóteses

Autores em [Weiss et al. 2007] propõem ilustrar o problema de violação de hipóteses. Neste exemplo, um cliente deseja comprar itens em uma aplicação de comercio eletrônico. Os requisitos desejados pelo cliente são definidos nos serviços abstratos $T1$ e $T2$, conforme Figura 2. Clientes selecionam itens para compra e solicitam tais itens a fornecedores. Os Serviços $S3$ e $S5$ são selecionados para cumprir os requisitos do cliente $T1$ e $T2$, respectivamente. Se $S5$ não tiver itens no seu estoque, encaminhará a solicitação de itens ao $S4$. Se ocorrer o mesmo com $S4$, este encaminhará a solicitação de itens a $S8$, e se $S8$ não tiver itens, este encaminhará a solicitação de itens a $S5$, formando assim um laço infinito de solicitações.

O laço infinito viola o requisito funcional *fornecer itens*, de $S5$ e requisitos não funcionais atribuídos a $S5$, por exemplo, tempo de resposta. Uma possível solução para esta interação de características seria substituir um serviço $S5$ por um serviço similar que não encaminhe solicitações de itens para outro fornecedor.

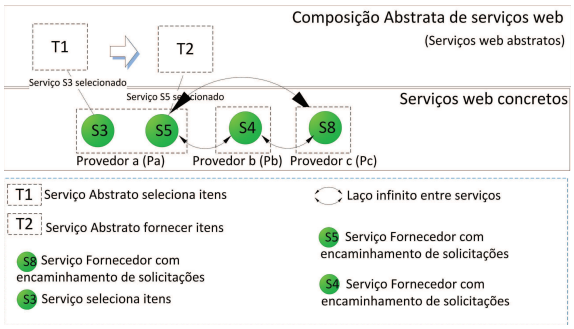

Figura 2. Laço infinito entre Serviços Web

3.3. Conflito entre metas

Nesse cenário, ilustrado por autores em [Weiss et al. 2007], três serviços participam da interação. O serviço identidade, o serviço fornecedor e o serviço cliente.

O serviço cliente tem como requisito funcional solicitar pedidos de itens para compra e visualizar o andamento dos seus pedidos. O serviço cliente também, por conveniencia, delega o gerenciamento do seu perfil a outro serviço. Este perfil é constituído de informações de endereço residencial, telefone, nome completo, CPF, dentre outros. No entanto, o serviço cliente deseja que a privacidade nos dados do seu perfil seja mantida, de modo que alguns serviços não tenham acesso à tais informações.

O serviço identidade é um serviço de gerenciamento de identidade. Ele permite a autenticação de serviços cliente que desejam acessar um serviço fornecedor. A autenticação tem como objetivo validar o serviço cliente de modo que este possa interagir com o serviço Fornecedor. Além disso, o serviço identidade realiza o gerenciamento do perfil do serviço cliente, armazenando os dados do perfil do serviço cliente em seus recursos e disponibilizando o perfil para que outros serviços o acesse.

O serviço Fornecedor é responsável por obter um pedido de compra de itens do serviço cliente e verificar a disponibilidade destes itens em seu estoque. Se existirem os itens no estoque, o serviço fornecedor prepara os itens para entrega. Para isto, o serviço fornecedor utiliza as preferencias de entrega do serviço cliente. Tais preferências estão armazenadas nos recursos do serviço identidade e representam as informações do perfil do cliente. Desta forma, o fornecedor solicita ao serviço identidade o acesso ao perfil do cliente. Além disso, o fornecedor possui o requisito de autenticar clientes. As operações necessárias para o alcance de tal requisito são delegadas ao serviço identidade.

O serviço cliente utiliza o serviço concreto identidade para alcançar os seguintes requisitos funcionais e não funcionais: “Gerenciar perfil”, “autentique usuário”,“privacidade” e “conveniencia”.

O serviço fornecedor, também utiliza o serviço concreto identidade para alcançar os seus requisitos funcionais e não funcionais. Neste exemplo, tais requisitos são: “Autenticar usuário” e “Acessar perfil usuário”.

Uma interação de características pode ocorrer se um ou mais requistos dos serviços não são alcançados. No cenário da figura 3, o requisito “privacidade” do serviço cliente pode não ser alcançado se o cliente não desejar que determinado fornecedor acesse

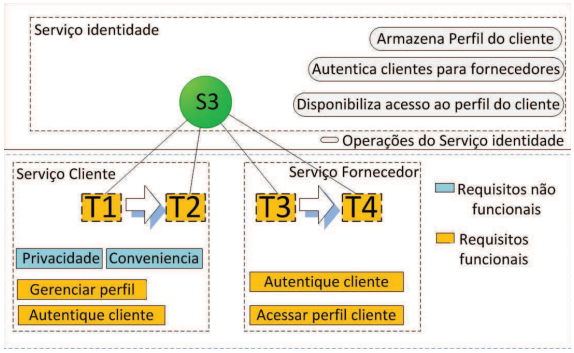

Figura 3. Violação do requisito do usuário devido a um conflito entre metas

o seu perfil, e no entanto, o serviço identidade permitir que determinado serviço fornecedor acesse esse perfil.

4. Trabalhos relacionados

Diferentes domínios abordam o problema da interação de característica. Autores em [Wang et al. 2009] propuseram mecanismos de detecção no domínio de telecomunicações. [Nakamura et al. 2009] trataram da detecção no domínio de uma rede doméstica, no entanto, o domínio Web é caracterizado por ser mais dinâmico e heterogêneo, portanto, uma abordagem diferente faz-se necessária.

4.1. Prevenção

A etapa de prevenção no domínio Web ocorre antes da execução de um serviço. Técnicas são propostas no intuito de prevenir a ocorrência de interação de características. As técnicas de prevenção são aplicadas a problemas relacionados a projetos inadequados e deficiência de plataforma ou arquitetura [Cameron e Velthuijsen 1993]. A prevenção pode ser definida como uma etapa que tem como meta solucionar as causas de uma interação de características de modo que esta não ocorra ou seja minimizada na composição.

Autores em [Kang et al. 2011], propuseram um algoritmo para seleção de serviços baseado no QoS das requisições dos usuários. Um conjunto de requisições são consideradas para decidir qual o melhor serviço para cada requisição. Este mecanismo analisa interação de características durante o processo de otimização. Autores em[Ai e Tang 2008], utilizaram um algoritmo genético para seleção baseado em penalidades para tratar o problema do QoS na composição de serviços web, considerando restrições tais como dependencias e conflitos. Este método analisa também as possíveis interações de características durante o processo de otimização.

4.2. Detecção

Nem todas as interações de características no domínio Web podem ser prevenidas. Devido a imprevisibilidade dos serviços que poderão participar de uma composição em tempo de execução, novas interações de características poderão ocorrer, sendo necessário então detectá-las. A detecção de interações de características identifica através de técnicas se interações de características indesejáveis estão presentes entre características

[Cameron e Velthuijsen 1993]. Alguns trabalhos propõem técnicas *online*(tempo de execução, no entanto, técnicas off-line são utilizadas para análise dos casos de interações.

Autores em [Weiss e Esfandiari 2004] propuseram técnica *offline* para detectar interações de características. A técnica consiste em modelar interação de características por um grafo, representando o relacionamento entre características. A análise desse modelo de modo manual foi realizada de modo que a interação de característica fosse detectada. Para eliminar as interações de características encontradas, o grafo de meta foi reprojetado. Autores em [Luo et al. 2009] descrevem também uma solução *offline*. Código BPEL4WS é traduzido para o formalismo timed automata e então o motor de model checking UPPAAL detecta interações de características. Tipos de interações de características como deadlock, situação de looping, erro de invocação e condição de disputa são representados como propriedades em um lógica temporal CTL e então verificadas pelo UPPAAL. Autores em [Zhang et al. 2006] modelaram uma composição de serviços através de redes de petri para detectar condição de disputa. Um algoritmo constroi uma árvore de alcançabilidade através de pares de transições concorrentes. Autores em [Hu et al. 2010] propõe método off-line para detecção baseado em dados semânticos. Ontologias foram utilizadas para modelar features e permitir o raciocínio das interações de características. O SWRL foi utilizado para raciocinar na ontologia e detectar as interações.

Poucos trabalhos no domínio de serviços web tratam a detecção *online* de interações de características. Na etapa de detecção, as soluções consistem no uso de ontologias e raciocínio lógico.

Autores em [Xu et al. 2010], propuseram um detector de interações on-line que intercepta mensagens SOAP de um motor de execução de composição de serviços e extrai informação semântica das mensagens. Um módulo externo ao detector denominado gerente de regras extrai dados semânticos de arquivos OWL-s e recebe os dados semânticos extraídos das mensagens SOAP. O detector então instancia regras armazenadas no gerente e define predicados que representam o estado do serviço web. Por fim, o detector analisa se algum conflito existe no estado corrente do serviço. Um framework e método para detectar interações no domínio dos serviços web em tempo de execução foi proposto tendo como propriedades ser efetivo e evitar a exploração do espaço de estados. Autores em [Xu et al. 2011] propõe uma técnica on-line baseada em STRIPS para raciocinar com SWRL e OWL-s . Essa técnica analisa a semântica das mensagens SOAP que são interceptadas de um motor de execução. Informações semânticas tais como entrada, saída, pré-condição e efeito são extraídas das mensagens SOAP e arquivos OWL-s. Fatos e regras definem uma base de conhecimento que é consultada para detectar a interação.

Tabela 1. Propostas para detectar, resolver e prevenir interações de características

Propostas	Detecção	Prevenção
[Weiss and Esfandiari 2004]	*offline*	X
[Luo et al. 2009],[Zhang et al. 2006]	*offline*	X
[Hu et al. 2010]	*offline*	X
[Xu et al. 2011, Xu et al. 2010]	*online*	X
[Ai and Tang 2008, Kang et al. 2011]	X	*offline*
QIC autonômico	*online*	*online*

Diante dos dados apresentados na Tabela 1 é possível constatar que as técnicas

de prevenção propostas são caracterizadas como *offline* e que as técnicas de detecção *online* não previnem a ocorrência de interação de característica. Desta forma, na próxima seção será descrita a proposta deste artigo que consiste em uma técnica de prevenção *online*, denominada QIC autonômico, baseada no histórico de detecções de interação de característica.

5. Propondo a prevenção através do histórico de detecções

Diante dos exemplos apresentados na seção anterior, observa-se que é possível prevenir ou minimizar interações de características na composição web através da seleção de serviços concretos que não ocasionem efeitos colaterais na composição de Serviços Web. Para isso, uma técnica de otimização que selecione os melhores serviços baseado em uma medida de qualidade de interação de características(QIC) foi proposta. Esta medida pode ser obtida através do monitoramento em tempo de execução dos serviços concretos, no intuito de detectar interação de características, analisar possíveis causas da violação e computar a medida de qualidade (QIC) para cada serviço concreto.

Devido a dinâmica, heterogeneidade e abertura dos serviços web, o monitoramento em tempo de execução dos serviços concretos deve ser realizado por mecanismos de auto gerenciamento. Assim, o presente trabalho propõe gestores autonômicos para monitoramento.

5.1. Computação autonômica

Computação autonômica é um paradigma destinado a incorporar mecanismos de auto-gerenciamento em sistemas computacionais[Tianfield e Unland 2004]. Autores em [Kephart e Chess 2003] definem um sistema autonômico como um sistema computacional capaz de se auto-gerenciar, tendo como referência metas de alto nível definidas por um administrador.

Sistemas autonômicos são projetados por elementos autonômicos. Cada elemento autonômico consiste das seguintes partes [Parashar e Hariri 2005]:

- Elemento gerenciado: menor unidade funcional de uma aplicação.
- Controle: Um gerente autonômico para cada elemento gerenciado.
- Ambiente: Todos os objetos que impactam no elemento gerenciado. Qualquer mudança nesses objetos pode causar o estado inconsistente da aplicação.

Autores em [Kephart e Chess 2003] definem um gerente autonômico como uma entidade que controla partes do sistema denominadas de elemento gerenciado através de um ciclo de gerenciamento denominado MAPE-BC (Monitorar, Analisar, Planejar, Executar, e Base de Conhecimento). O monitor recupera os dados capturados do ambiente gerenciado e realiza diversos tratamentos necessários para posterior análise. O analisador busca informações de modo a aprender sobre aspectos do ambiente gerenciado e consequentemente prever alguns possíveis comportamentos. O planejador utiliza políticas para decidir dentre diversas alternativas de execução, a melhor. Enfim, o executor realiza o controle da execução de um plano.

5.2. Gestores autonômicos

Neste trabalho, um gerente autonômico foi proposto. O gerente monitora mensagens que são enviadas para o Serviço Web monitorado e através de uma Rede Neural tenta

encontrar um padrão que represente uma causa de interação de características. Após detecção, um analisador de qualidade atribui um valor de qualidade para cada serviço concreto avaliado. Diante dos dados de qualidade aplicados a cada Serviço Web, um algoritmo genético otimizará a composição de modo que novos serviços concretos sejam associados a cada serviço abstrato.

5.2.1. Redes neurais autonômicas para detecção de interação de características

Os gerentes autonômicos detectam interações de características através da utilização de uma rede neural. Mensagens trocadas entre o serviço concreto e o serviço cliente são interceptadas no intuito de detectar padrões de interações de características entre dois serviços de uma composição. Basicamente, a interação entre cliente e serviço concreto ocorre da seguinte forma: o serviço cliente envia uma mensagem para o Serviço concreto, passando como parâmetro uma entrada I. Assim, o Serviço concreto retorna uma saída O. Estes dados caracterizam o requisito funcional do Serviço concreto. Além disso, requisitos não funcionais podem ser representados no serviço concreto. Dentre eles, o tempo de resposta, privacidade, conveniência e recurso disponível. Assim, os dados necessários para analisar uma interação entre um cliente e o serviço concreto foram definidos por $int =< I, O, TR_d, TR_o, P, C, R >$, onde tais simbolos representam entrada (I), saida (O), tempo de resposta desejado (TR_d), tempo de resposta obtido (TR_o), privacidade (P), conveniência (C) e recurso disponível (R). Para detectar interações de características, duas interações int_1 e int_2 entre dois clientes e o serviço concreto são necessárias. Desta forma, dados foram apresentados à rede neural para classificação, no seguinte formato: $< int_1, int_2 >$, no qual int_1 é a interação entre o cliente 1 e o Serviço concreto e int_2 é a interação entre o cliente 2 e o mesmo Serviço concreto.

Os resultados obtidos foram baseados em três amostras de padrões, sendo que cada amostra contem doze padrões. Para cada padrão, uma rede neural *Backpropagation*, de quatorze entradas, duzentos neurônios na camada oculta e 2 neurônios na camada de saída foi utilizada para classificar os padrões em 4 classes: contenção de recursos, violação de hipoteses, conflito entre metas e situação livre de interações de características. O *toolbox* de redes neurais ([e Mark Beale 2000]) da ferramenta MATLAB foi utilizado para configurar a rede. A Tabela 1 mostra as precisões e tempo de processamento da rede neural para cada amostra apresentada.

Tabela 2. Desempenho e precisão na detecção das causas de interações de características por RNA

Dados de entrada	Tempo detecção	precisão
amostra 1	0,020157	83,3
amostra 2	0,020561	91,7
amostra 3	0,020000	91,7

Após detecção, o gerente autonômico classifica os serviços concretos baseado na frequência de detecções realizadas pela rede neural. Uma medida de qualidade (QIC) é calculada de acordo com a formula: $QIC_{classe} = nocorrencias \div tempo$, onde:

$nocorrencias$ é a quantidade de interações ocorridas para uma determinada classe; $tempo$ representa o tempo decorrido por hora. Um valor médio é então calculado entre as classes de interação de características, $QIC = (QIC_{cr} + QIC_{vh} + QIC_{cm}) \div 3$, onde QIC_{cr}, QIC_{vh} e QIC_{cm} correspondem às classes de contenção de recurso, violação de hipóteses e conflito entre metas, respectivamente. O valor QIC, compreendido na faixa de valores entre $[0, 1]$ é então atribuído ao serviço concreto.

5.3. Prevenção de interação de características

Este experimento analisou a viabilidade de otimizar uma medida de QIC. A ferramenta GA solver ([Mathworks 2011]) foi utilizada no MATLAB, no intuito de selecionar os melhores Serviços Web, com a menor probabilidade de ocasionar interação de característica.

Foi definida uma população de tamanho = 10 de indivíduos, onde os indivíduos são cromossomos com 6 genes. A faixa de valores de QIC de cada gene foi entre [0,1]. O algoritmo genético então utilizou os operadores de mutação e crossover padrões do Ga solver para evoluir a população de cromossomos. A mutação gaussiana modificou o gene do cromossomo, considerando os limites superiores e inferiores dos possíveis valores de aptidão estabelecidos. O operador de crossover escolheu um par de cromossomos pais e para cada gene dos cromossomos pais, na mesma posição, foi escolhido randomicamente o gene que seria transmitido ao filho.

A Tabela 3 mostra as soluções encontradas para um fluxo de seis serviços abstratos $T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 \rightarrow T5 \rightarrow T6$. Cada serviço abstrato tem disponível infinitos serviços concretos para seleção, cada um destes, representados por seus respectivos valores de QIC. Nos resultados da tabela, um serviço concreto para cada serviço abstrato é selecionado, caracterizando uma solução S. Cada solução representa uma execução distinta do algoritmo genético.

Tabela 3. Otimização dos serviços concretos da composição abstrata por Algorítmos genéticos

S	T_1	T_2	T_3	T_4	T_5	T_6	Tempo
1	0	0	0,36	0,27	0	0	0,25
2	0,24	0	0,04	0	0,06	0,12	0,33
3	0	0,023	0	0,10	0	0,023	0,55
4	0,02	0	0,87	0,15	0	0,54	0,32
5	0,15	0,06	0,193	0,02	0,08	0,53	0,44

5.4. Discussão

As técnicas *offline* são insuficientes para solucionar interações de características, pois mudanças podem ocorrer na configuração da composição uma nova interação de características ocorrer. Assim, Técnicas *online* são fundamentais para solucionar interações de características na composição de serviços web.

Diante dos dados encontrados, foi constatado que a maioria das técnicas para prevenir e detectar interações de características no domínio de serviços web são realizadas *offline* e poucas abordagens *online* foram encontradas nesta revisão.

Na etapa de detecção, as técnicas *offline* baseadas em métodos formais possuem a limitação de verificar todos os possíveis estados dos serviços pertencentes a uma composição. Isso acarreta no problema de explosão do espaço de estados, dificultando no desempenho do método.

A comparação entre os trabalhos correlatos e a proposta deste trabalho, conforme Tabela 1, demonstra que a principal contribuição desta proposta é implementar um mecanismo de prevenção *online* através da otimização de uma medida de QIC. Em outras abordagens, conforme autores em [Ai e Tang 2008, Kang et al. 2011], a análise de interações indesejáveis entre serviços ocorre de modo *offline*.

Diante dos dados apresentados na Tabela 2 foi constatado que uma rede neural pode ser uma técnica adequada para detectar interações de características em laço de controle autonômico devido ao baixo tempo de detecção da rede, que detecta doze padrões de interação entre serviços em uma média de 0.020 segundos. Além disso, a precisão média da rede, na faixa de 88.9 demonstra bons resultados para a deteção de interações em tempo de execução.

Diante dos dados obtidos conforme Tabela 3 foi constatado que o algorítmo genético pode otimizar uma medida de QIC, pois os resultados para cada serviço abstrato mostra o QIC dos serviços concretos próximo de zero. Valores próximos de zero representam um bom QIC e valores próximos de 1 representam um QIC ruim. .No entanto, o tempo médio para encontrar uma solução corresponde a 0.37 segundos.

6. Conclusão

Este trabalho avaliou a possibilidade de solucionar interações de características na composição web. Foi constatado que uma solução *online* é fundamental para lidar com as características de um ambiente web dinâmico, heterogêneo e aberto.

Desta forma, uma solução *online* autonômica baseada em redes neurais e algoritmos genéticos para detectar e prevenir interações de características foi proposta. Diante dos resultados na etapa de detecção por redes neurais, evidências demonstram a precisão da técnica e a resposta rápida necessária para a reação rapida à mudanças. A prevenção através dos algoritmos genéticos mostra a viabilidade de otimização da composição baseada em uma medida de qualidade de interações de características. Assim, os resultados obtidos demonstram a viabilidade para soluções em ambientes dinâmicos.

Referências

Ai, L. and Tang, M. (2008). A penalty-based genetic algorithm for qos-aware web service composition with inter-service dependencies and conflicts. In *International Conferences on Computational Intelligence for Modelling*, pages 738–743.

Alonso, Gustavo e Casati, F. e. K. H. e. M. V. (2004). *Web services: Concepts, architectures and applications*. Data-centric systems and applications. Springer.

Amorim, R., Claro, D. B., Lopes, D., Albers, P., and Andrade, A. (2011). Improving web service discovery by a functional and structural approach. In *ICWS*, pages 411–418.

Calder, M., Kolberg, M., Magill, E. H., and Reiff-Marganiec, S. (2003). Feature interaction: a critical review and considered forecast. *Comput. Netw.*, pages 115–141.

Cameron, E. and Velthuijsen, H. (1993). Feature interactions in telecommunications systems. *Communications Magazine, IEEE*, pages 18 –23.

Crespo, R. G., Carvalho, M., and Logrippo, L. (2007). Distributed resolution of feature interactions for internet applications. *Comput. Netw.*, pages 382–397.

e Mark Beale, H. D. (2000). Neural network toolbox user guide. The MathWorks, Inc.

Hu, H., Yang, D., Fu, C., and Fang, W. (2010). Towards a semantic web based approach for feature interaction detection. In *Software Technology and Engineering (ICSTE), 2010 2nd International Conference on*, volume 2, pages V2–330 –V2–334.

Kang, G., Liu, J., Tang, M., Liu, X., and Fletcher, K. (2011). Web service selection for resolving conflicting service requests. In *Web Services (ICWS)*, pages 387 –394.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, pages 41–50.

Luo, X., Xuan, A., and Dong, R. (2009). Detecting feature interactions in web services with timed automata. In *Proceedings of the 2009 Third International Conference on Genetic and Evolutionary Computing*, pages 276–279. IEEE Computer Society.

Mathworks (2011). *Global Optimization Toolbox Documentation*.

Nakamura, M., Igaki, H., Yoshimura, Y., and Ikegami, K. (2009). Considering online feature interaction detection and resolution for integrated services in home network system. In *InternationalConference on Feature Interactions*, pages 191–206.

Nguyen, X. T. and Kowalczyk, R. (2006). An agent based qos conflict mediation framework for web services compositions. *Intelligent Agent Technology, IEEE / WIC / ACM International Conference on*, pages 522–528.

Parashar, M. and Hariri, S. (2005). Autonomic computing: An overview. In *Unconventional Programming Paradigms*, pages 247–259. Springer Verlag.

Tianfield, H. and Unland, R. (2004). Towards autonomic computing systems. *Engineering Applications of Artificial Intelligence*, 17:689 – 699.

Wang, L., Xu, J., and Reiff-Marganiec, S. (2009). Online detection of feature interactions of cpl services. In *ICFI*, pages 19–33. IOS Press.

Weiss, M. and Esfandiari, B. (2004). On feature interactions among web services. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 88 – 95.

Weiss, M., Esfandiari, B., and Luo, Y. (2007). Towards a classification of web service feature interactions. *Comput. Netw.*, pages 359–381.

Xu, J., e Youxiang Duan, K. C., and Reiff-Marganiec, S. (2011). Modeling business process of web services with an extended strips operations to detection feature interaction problems runtime. *International Conference on Web Services*, pages 516–523.

Xu, J., Yu, W., Chen, K., and Reiff-Marganiec, S. (2010). Web services feature interaction detection based on situation calculus. In *Proceedings of the 2010 6th World Congress on Services*, pages 213–220.

Zhang, J., Su, S., and Yang, F. (2006). Detecting race conditions in web services. In *Proceedings of the Advanced Int'l Conference on Telecommunications*, pages 184–.