Uma Linguagem de Workflow Para Composição de Web Services - LCWS

Luiz Alexandre Hiane da S. Maciel[1], **Edgar Toshiro Yano**[1]

[1]Divisão de Pós-Graduação – Instituto Tecnológico de Aeronáutica (ITA)
Praça Marechal Eduardo Gomes, 50 – Vila das Acácias
12228-900 – São José dos Campos – SP – Brasil

{luizh,yano}@ita.br

Abstract. *The Business process automation has been getting very important within organizations, because it may be seen as a technology of components which replaces or supplements handwork in an effort to eliminate nonvalue-adding human interventions in the process. The Web Services composition through workflow modeling languages is considered an emerging paradigm that enables integration of internal applications and applications which cross organizational boundaries. This work presents a workflow language that use Web Services as components, an architecture for a run-time environment to this language and the advantages of the utilization of this kind of technology.*
keywords: Workflow, Workflow Languages, Web Services, Service-oriented architecture, business process

Resumo. *A automação de processos de negócios vem ganhando importância nas organizações, pois é entendida como uma tecnologia de componentes que substitui ou suplementa processos manuais, buscando eliminar intervenções humanas que não adicionem valor ao processo. A composição de* Web Services *através de linguagens para modelagem* Workflow *é considerada um paradigma emergente que possibilita a integração de aplicações internas, bem como de aplicações que transpõem as fronteiras organizacionais. Este trabalho apresenta uma linguagem de* Workflow *que utiliza* Web Services *como componentes, uma arquitetura de um ambiente de execução para tal linguagem e as vantagens da utililização deste tipo de tecnologia.*
palavras-chave: Workflow, Linguagens Workflow, Web Services, Arquitetura Orientada a Serviços, Processos de Negócio

1. Introdução

Empresas para serem competitivas não podem mais estar centradas apenas em processos manuais, ou seja, processos que dependam de atividades de pessoas para serem realizados. Situação que é agravada se essas atividades forem repetitivas e exigirem trabalho de um grupo de pessoas que estão fisicamente distantes. Assim, muitas empresas estão reavaliando seus negócios de modo a se tornarem mais produtivas e efetivas, o que conseqüentemente, exige que aplicações existentes sejam modificadas e novas aplicações sejam desenvolvidas.

O escopo do presente artigo envolve o desenvolvimento de processos de negócio formados por uma composição de serviços que estejam disponíveis em uma rede de computadores.

Assim, a tecnologia *Workflow* é utilizada para coordenar as interações entre os *Web Services*, os quais de forma composta visam alcançar um objetivo comum dentro da organização. E os *Web Services* representam os passos lógicos que compõem o *Workflow*, ou seja, os passos que devem ser executados para que se alcançe um objetivo de negócio dentro da organização.

1.1. *Workflow* e *Web Services*

Uma tecnologia cuja utilização vem aumentando nos negócios numa grande variedade de empresas é o gerenciamento *Worklflow*, em português gerenciamento do fluxo de trabalho. Uma entidade que se destaca no estudo sobre *Workflow* e que vem desenvolvendo padrões e terminologias comuns sobre o assunto, é a *Workflow Management Coalition - WfMC*.

Segundo esta entidade [WfMC 1999], *Workflow* é a automação total ou parcial de um conjunto de atividades interligadas, que coletivamente alcançam um objetivo de negócio. Durante a execução destas atividades podem ocorrer trocas de documentos, informações ou tarefas entre os participantes do *Workflow* com a finalidade de realizar alguma ação. Esta automação é realizada de acordo com um conjunto de regras de procedimento.

A Internet vem se tornando uma plataforma comum global em que organizações e indivíduos se comunicam para a execução de várias atividades. Quanto mais surgem aplicações e ferramentas baseadas na Web, mais e mais serviços são disponibilizados através da Internet [Chiu et al. 2001]. Desta forma, surge a possibilidade de compor serviços externos existentes, dentro de um aplicação interna, o que estimula o reuso e rápido desenvolvimento. Esta possibilidade causou um grande impacto nos sistemas de gerenciamento de *Workflow*, uma vez que os *Workflow*s podem ser executados através da Internet.

A tecnologia *Web Service* está baseada na arquitetura orientada a serviços. A Figura 1 ilustra esta arquitetura, apresentando três papéis ou funções e três operações. Os três papéis são provedor de serviço, requerente de serviço e registro de serviços. Os objetos que são manipulados são o serviço e a descrição do serviço. As operações que são executadas pelos atores fazendo uso desses objetos são publicar, encontrar e conectar [Gottschalk et al. 2002]. *Web Service* é um componente baseado em padrões, bem definido e autônomo, que pode ser acessado via protocolos baseados na Web [BEA et al. 2002].

A composição de Web Services é um paradigma emergente que possibilita a integração tanto de aplicações internas, como de aplicações que transpõem as fronteiras organizacionais [Wohed et al. 2002]. Desse modo, linguagens e técnicas para composição de Web Services têm surgido e estão sendo continuamente aprimoradas com novas propostas de diferentes empresas e coalizões.

O problema estudado neste trabalho é a dificuldade de desenvolver processos de *Workflow* formados por *Web Services*, com a finalidade de aumentar os índices de reutilização desses *Web Services*. Outra dificuldade explorada diz respeito a obtenção de uma visão real, isto é, prática, funcional, de uma arquitetura para um ambiente que forneça suporte à execução desses processos de *Workflow* e que possua código aberto/livre.

Assim, os objetivos deste trabalho são:

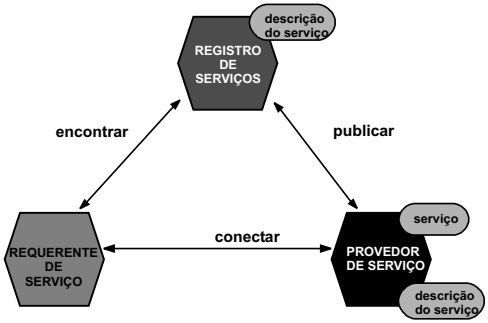

Figura 1. Atores, objetos e operações de *Web Services*. [Adaptado de [Gottschalk et al. 2002]]

1. Definir uma Linguagem Para Composição de *Web Services* (LCWS) com a finalidade de coordenar as interações entre os *Web Services* e manipular as informações necessárias para execução dos mesmos.
2. Desenvolver uma arquitetura para um ambiente de execução com a capacidade de interpretar um *Workflow* definido em tal linguagem e realizar as chamadas aos serviços na ordem que foi definida no *Workflow*.

As Seções deste trabalho estão organizadas da seguinte forma: A Seção 2. apresenta a Linguagem de *Workflow* para Composição de *Web Services* - LCWS. A Seção 3. apresenta o Ambiente de Execução para LCWS - AELCWS. A Seção 4. traz uma análise dos resultados obtidos com a utilização da LCWS e do AELCWS. A Seção 5. são apresentadas algumas conclusões, bem como propostas para trabalhos futuros.

2. Linguagem de *Workflow* Para Composição de *Web Services* - LCWS

Nesta seção é apresentada uma linguagem para representação de *Workflow* em XML. Para se definir tal linguagem foram considerados alguns dos padrões *Workflow* discutidos em [van der Aalst et al. 2000, van der Aalst et al. 2002, van der Aalst 2004].

A linguagem proposta neste trabalho faz parte do grupo dos *workflow* estruturados [Kiepuszewski et al. 2000]. Dessa forma, são definidas algumas estruturas para serem utilizadas na representação dos *workflows*. Os elementos devem estar contidos uns dentro dos outros, de forma análoga a uma estrutura hierárquica, o que facilita a sua compreensão.

Num *workflow* representado na linguagem proposta, as requisições por serviço são especificadas nas atividades, levando-se em consideração as informações obtidas na descrição de cada *Web Service*. Durante a execução, tais atividades são conectadas com o provedor de serviço que foi especificado na modelagem.

A seguir os elementos da linguagem LCWS são sucintamente apresentados. Esses elementos foram escolhidos, pois fornecem suporte as construções básicas para uma linguagem *Workflow*. Essas construções básicas por sua vez fornecem suporte a perspectiva de controle de fluxo, considerada essencial para a efetividade de uma especificação *Workflow*. Esta perspectiva descreve atividades e sua ordem de execução através de dife-

rentes construtores, os quais permitem um fluxo para controle de execução, por exemplo, seqüência, divisões, paralelismo e junção sincronizada [van der Aalst et al. 2002].

2.1. Elementos da LCWS

Nesta Subseção serão descritos os elementos que compõem a linguagem proposta, de uma forma sucinta. No texto, serão escritos da seguinte forma: **<elemento>**, em que "elemento" corresponde ao nome do elemento da linguagem.

A linguagem contempla duas perspectivas importantes em um *Workflow*, a saber: perspectiva de controle de fluxo e perspectiva de dados. Esta segunda perspectiva representa as informações passadas e geradas durante a execução do *workflow*. É possível passar informações ao se iniciar uma instância do *workflow* através de um contexto inicial. Já a manipulação dos dados gerados durante a execução da instância é realizada através de variáveis declaradas dentro do *workflow*. Cabe ressaltar que a ênfase maior pertence ao controle de fluxo, visto que este é o maior responsável pela ordem de execução das atividades do *workflow*. Dessa forma, limitou-se o escopo da linguagem ao que é essencial, com o intuito de prover um ambiente de execução para a LCWS.

A definição da LCWS foi realizada através da linguagem *XML Schema*. A Figura 2 fornece uma visão geral desse *XML Schema* que define a LCWS. Os retângulos com linhas cheias representam os elementos que têm presença obrigatória, já os retângulos com linhas tracejadas, indicam os elementos que possuem presença facultativa.

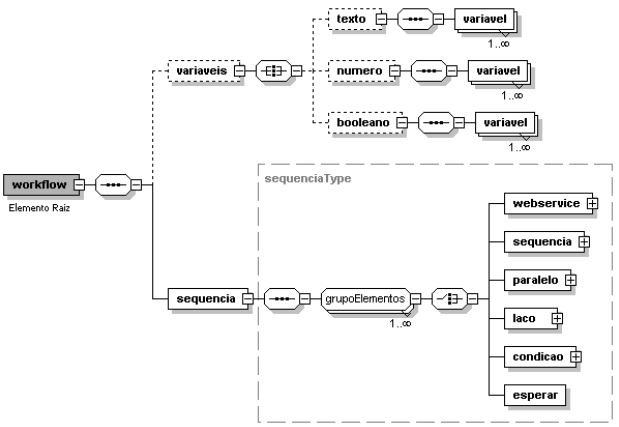

Figura 2. Visão geral do *XML Schema* para a LCWS

Na Figura 2 existem os indicadores de ordem da linguagem *XML Schema*: *sequence, all e choice*. Por exemplo, após o elemento **<workflow>** temos um indicador *sequence*, após o elemento **<variaveis>** temos o indicador *all* e após o grupo `grupoElementos` temos o indicador *choice*. Os indicadores de ordem são utilizados para definir como os elementos, no caso os elementos da linguagem LCWS, podem ocorrer dentro do arquivo XML [W3Schools 2004].

Os elementos e estruturas válidos para a linguagem proposta, ilustrados na Figura 2, são detalhados a seguir.

1. **Web Service - `<webservice>`**: O elemento **`<webservice>`** é o responsável pela execução da atividade propriamente dita, representando dessa maneira a menor unidade de trabalho da linguagem. É o elemento que invoca os *Web Services* que fazem parte da instância de *workflow* que está sendo executada.
   Os demais elementos da linguagem, que representam as estruturas predefinidas da linguagem, são responsáveis pelo direcionamento do fluxo do *workflow*, ou seja, determinam a rota a ser seguida na execução dos *Web Services* que compõem a instância de *workflow*.
   A Figura 3 expande o elemento **`<webservice>`** da Figura 2. Fornece uma visão geral da estrutura do elemento **`<webservice>`**, como definida no *XML Schema* da LCWS.

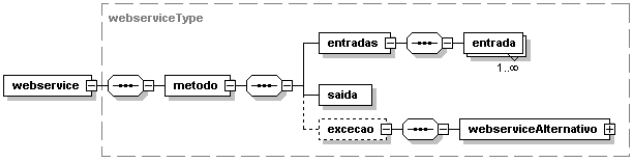

Figura 3. Elemento `<webservice>`

   As principais informações necessárias para configurar este elemento são: o URL em que o *Web Service* está disponível, o nome do serviço, o name space e a porta do mesmo. Informa-se o método a ser invocado e os parâmetros de entrada para a execução. Por fim, informa-se o tipo e nome da saída, ou seja, da resposta fornecida após a execução do Web Service.
   Existe mais uma informação que pode estar contida nesse elemento, porém não possui presença obrigatória, a qual seria um *Web Service* alternativo que deve ser executado caso ocorra alguma exceção na execução do *Web Service* atual. O *Web Service* alternativo é estruturado da mesma forma que o elemento **`<webservice>`**.
2. **Workflow - `<workflow>`**: O elemento **`<workflow>`** é o elemento raiz do documento XML, de forma que engloba todos os demais elementos, como ilustrada na Figura 2. Possui dois elementos filhos imediatos, **`<variaveis>`** e **`<sequencia>`**, sendo que o segundo tem presença obrigatória.
3. **Variáveis - `<variaveis>`**: O elemento **`<variaveis>`** centraliza a declaração das variáveis que serão utilizadas na execução do *workflow*. Fornece suporte ao fluxo de dados, que representa as informações passadas como contexto inicial mais as informações geradas durante a execução do *Workflow*.
   Sua presença é facultativa dentro do elemento **`<workflow>`**.
   Possui três filhos imediatos, a saber, **`<numero>`**, **`<texto>`** e **`<booleano>`**, que servem para indicar o tipo de variável que está sendo declarada. Desta forma, a linguagem proposta suporta três tipos de variáveis, que podem ser manipuladas durante a execução de uma instância do *workflow*.
   Cada um desses três elementos pode conter um ou mais elementos **`<variavel>`**, que deve conter um nome e um valor inicial.

4. **Seqüência - `<sequencia>`**: Dentro deste elemento, podem estar presentes qualquer um dos outros elementos pertencentes a linguagem, os quais podem aparecer uma ou mais vezes. Os elementos **`<workflow>`** e **`<variaveis>`**, são os únicos que não podem estar contidos nesse elemento.
   Os elementos contidos dentro de **`<sequencia>`** serão processados um após o outro. Um elemento de um processo *Workflow* é habilitado para execução somente após o término de um outro elemento no mesmo processo, ou seja, após a finalização de seu antecessor.
5. **Paralelo - `<paralelo>`**: O elemento **`<paralelo>`** representa a estrutura responsável por executar simultaneamente os elementos que estão em seu interior. Pode conter os mesmos elementos que **`<sequencia>`**. A diferença está no fato de que seus filhos serão executados em paralelo, ou seja, ao mesmo tempo. Este elemento deve possuir no mínimo dois elementos filhos.
   Cabe ressaltar, que esse paralelismo possui sincronismo, de maneira que para prosseguir na execução do fluxo de trabalho faz-se necessário o término de todos os fluxos filhos que estejam executando em paralelo.
6. **Condição - `<condicao>`**: Representa a estrutura na qual uma escolha é realizada, para seguir um dos caminhos disponíveis no fluxo do *workflow*, durante a execução de uma instância do mesmo.
   No elemento **`<condicao>`** duas ramificações estão disponíveis, representadas pelos elementos **`<verdadeiro>`** e **`<falso>`**, os quais habilitam o caminho a seguir. Podem envolver apenas um dos outros elementos da linguagem, exceto **`<worfklow>`** e **`<variaveis>`**. No caso de condição ter sido avaliada como verdadeira, segue-se a elemento **`<verdadeiro>`** e vice-versa.
7. **Laço - `<laco>`** O elemento **`<laco>`** representa um laço de repetição. O conteúdo de **`<laco>`** é constituído de dois elementos filhos: **`<condicao>`**, que apesar de possuir os mesmos atributos, não é o descrito anteriormente, uma vez que diz respeito apenas à condição do laço; o segundo elemento pode ser um dos demais elementos da linguagem, exceto **`<workflow>`** e **`<variaveis>`**. O segundo elemento filho é executado enquanto a condição for avaliada como verdadeira.
8. **Esperar - `<esperar>`** O elemento **`<esperar>`** não possui filhos. Um elemento **`<esperar>`** é utilizado para determinar pontos no fluxo do processo em que a instância em execução deve esperar por um determinado tempo.

2.1.1. Representação de Outros Padrões *Workflow*

A linguagem proposta fornece suporte aos padrões: **seqüência**, **ramificação paralela**, **sincronização**, **escolha exclusiva**, **união simples** e **iteração**. Os demais padrões *Wokflow* discutidos em [van der Aalst et al. 2002, van der Aalst et al. 2000] podem ser suportados indiretamente através da combinação dos elementos apresentados, ou então não são suportados pela linguagem.

A Figura 4 apresenta uma visão geral da relação existente entre o registro de serviços, o provedor de serviço, o modelador e o *workflow* modelado com a LCWS. O modelador pode buscar em um registro de serviços os *Web Services* necessários para modelagem do processo de negócio; em seguida pode especificar as informações necessárias

nas atividades do *Workflow* que representa o processo. Essas atividades se conectam, em tempo de execução, com os respectivos provedores de serviços.

Nessa Figura os *Web Services* são representados por caixas com linhas preenchidas, sendo que cada letra representa um *Web Service* diferente. Os círculos preenchidos representam ramificações e junções em que apenas uma das linhas de controle é executada, constituindo ramificações condicionais que atendem aos padrões **escolha exclusiva** e **união simples** (`<condicao>`), enquanto que os círculos vazios indicam ramificações e junções em que todas as linhas de controle serão executadas simultaneamente, constituindo ramificações paralelas que atendem aos padrões **ramificação paralela** e **sincronização** (`<paralelo>`).

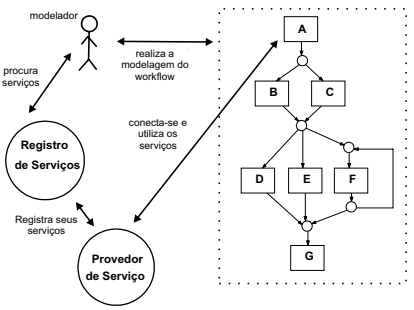

Figura 4. Interação entre o Registro de serviços, o Provedor de Serviço, o modelador, e o worklfow.

A Figura 5 fornece um exemplo de como o *Workflow*, da Figura 4, ficaria modelado na LCWS. Note que algumas informações foram omitidas, para se facilitar o entendimento.

```
1  <workflow>
2     <sequencia>
3        <webservice nome="A"...>
4           ...
5        </webservice>
6        <condicao ...>
7           <verdadeiro>
8              <webservice nome="B" ...>
9                 ...
10             </webservice>
11          </verdadeiro>
12          <falso>
13             <webservice nome="C" ...>
14                ...
15             </webservice>
16          </falso>
17       </condicao>
18       <webservice nome="D" ...>
19          ...
20       </webservice>
21       <paralelo>
22          <webservice nome="E" ...>
23             ...
24          </webservice>
25          <webservice nome="F" ...>
26             ...
27          </webservice>
28          <laco ...>
29             <condicao .../>
30             <webservice nome="G" ...>
31                ...
32             </webservice>
33          </laco>
34       </paralelo>
35       <webservice nome="H" ...>
36          ...
37       </webservice>
38    </sequencia>
39 </workflow>
```

Figura 5. Exemplo Geral de modelagem na LCWS

2.2. Considerações sobre a LCWS e trabalhos relacionados

Os padrões de controle escolhidos para serem suportados pela LCWS foram selecionados com base na constatação de que a maioria das linguagens que fornecem suporte a *Workflow*, que são consideradas trabalhos correlatos, como, XPDL [WfMC 2002],

XLANG [Microsoft 2001], WSFL [IBM 2001], BPEL4WS [BEA et al. 2003] e WSCI [BEA et al. 2002] possuem suporte direto aos seguintes padrões: seqüência, ramificação paralela, sincronização, escolha exclusiva e união simples. Já o padrão iteração, também conhecido como ciclos arbitrários, possui suporte direto apenas na linguagem XPDL. As demais linguagens permitem apenas ciclos estruturados, ou seja, podem possuir apenas um único ponto de entrada e um único ponto de saída do ciclo.

Optou-se por implementar uma linguagem própria, a LCWS, com o intuito de se conseguir uma maior liberdade para definição dos elementos internos da linguagem. Essa escolha é considerada importante, pois um dos objetivos deste trabalho é o de disponibilizar além da linguagem uma arquitetura para um ambiente de execução que forneça suporte a linguagem proposta.

Assim, a LCWS é uma linguagem que fornece suporte a padrões considerados fundamentais para linguagens de *Workflow*. É uma linguagem bem definida e com o escopo bem delimitado, que é suportada pela arquitetura que será melhor detalhada na Seção 3..

3. Ambiente de execução

Para fornecer suporte à linguagem proposta para composição de *Web Services* (LCWS), foi desenvolvido um ambiente de execução capaz de interpretar e executar processos definidos através da mesma. Neste Seção, discute-se a arquitetura deste ambiente desenvolvido de forma sucinta, mostrando as principais relações entre este e o modelo de referência da *WfMC*. O ambiente de execução para LCWS será chamado, para facilitar a escrita, de Ambiente de Execução para Linguagem de *Workflow* para Composição de *Web Services* (AELCWS).

3.1. Terminologia para o AELCWS

A terminologia adotada é muito parecida com a que é adotada pela WfMC. A grande diferença está no fato do AELCWS não fornecer suporte a atividades que necessitem da interação com pessoas, uma vez que todas as atividades são realizadas de forma automatizada através dos serviços que devem estar disponíveis através de uma rede de computadores, os chamados *Web Services*. A Figura 6 ilustra a terminologia adotada.

Observa-se, nesta Figura, que uma definição do processo de negócio é criada, ou seja, uma representação do que se espera acontecer, baseada no próprio processo de negócio. Essa definição composta por atividades, os *Web Services*, é utilizada pelo Sistema de Gerência de *Workflow* para controlar o processo de negócio. Esse controle é realizado utilizando-se instâncias do processo, ou seja, representações do que está acontecendo na prática. As atividades automatizadas, que fazem parte da definição do processo de negócio, são representadas por instâncias de atividades em tempo de execução, as quais são executadas de maneira automática através de *Web Services*.

3.2. Modelo de Referência da WfMC e AELCWS

Existem 5 interfaces entre um Sistema de Gerência de *Workflow* e o ambiente no qual está envolvido, as quais foram propostas pela WfMC como modelo de referência para *Workflow*, com o intuito de disponibilizar uma representação da arquitetura de um Sistema de Gerência de *Workflow* [WfMC 1999]. A relação do ambiente AELCWS e tais interfaces encontra-se ilustrada na Figura 7 e é detalhada a seguir.

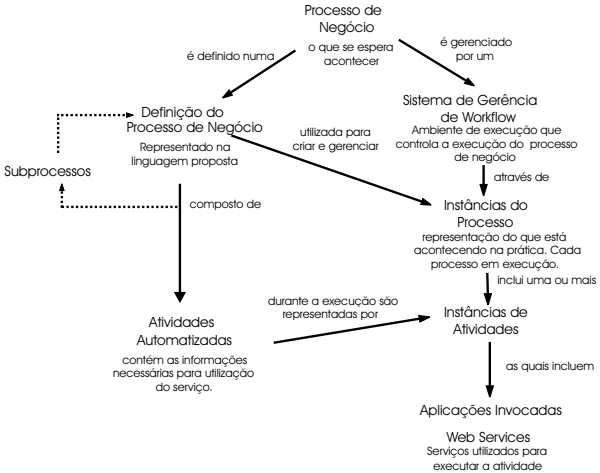

Figura 6. Terminologia adaptada da proposta da WfMC, para o AELCWS.

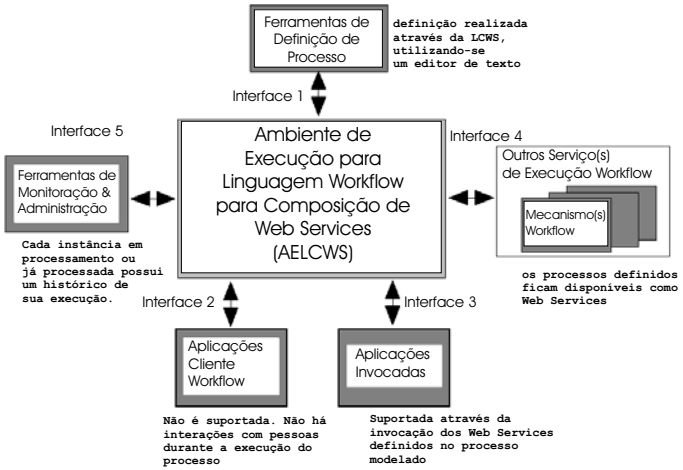

Figura 7. Relação entre o AELCWS e as interfaces propostas pela WfMC.

Interface 1 – É a interface entre o ambiente de modelagem e o serviço de execução da arquitetura. É suportada pelo AELCWS fazendo uso da linguagem LCWS para modelar os *Workflow*s que definem processos e informações relacionadas. Já a interpretação e a execução desse processo são realizadas através do ambiente AELCWS, o qual é responsável por executar as atividades, ou seja, os *Web Services*, na ordem imposta na definição do processo.

Interface 2 – É a interface que realiza as interações entre o serviço de execução

e os participantes que interagem dentro de um processo de negócio. O AELCWS não fornece suporte à interação direta com pessoas durante a execução do processo, de modo que essa interface não é implementada.

Interface 3 – É a interface responsável pelas interações entre o serviço de execução e as aplicações que não necessitam da intervenção de pessoas. O AELCWS fornece suporte a esta interface através da invocação dos serviços que foram escolhidos na definição do processo, em que são realizadas conexões com os provedores desses serviços, em tempo de execução, para habilitar a utilização dos mesmos.

Interface 4 – É a interface entre dois serviços de execução distintos. O suporte a esta interface não é considerado completo, porém pode ser realizado através da característica de que um processo definido através da LCWS fica disponível como um *Web Service* comum, permitindo assim que outros serviços de execução possam utilizá-lo sem grandes complicações. Existe uma restrição nesse caso, a saber, a passagem de parâmetros de contexto inicial ainda não é possível. É permitido apenas utilizar outros processos que não necessitem de um contexto inicial para serem executados.

Interface 5 – É a interface entre o serviço de execução e o ambiente de administração desse serviço. Esta interface também é parcialmente suportada pelo AELCWS, uma vez que este ambiente fornece um histórico para cada instância de processo *Workflow* que esteja em execução ou que já foi terminada. Assim, é possível verificar como está sendo ou foi a execução da instância do processo, obtendo, por exemplo, informações manipuladas e retornos dos *Web Services* invocados.

3.3. Implementação do AELCWS

Nesta Subseção detalha-se o AELCWS, mostrando seu funcionamento interno. A Figura 8 ilustra a relação entre os elementos internos do ambiente. Cabe ressaltar que todos os elementos internos do AELCWS foram implementados neste trabalho.

A seguir, descreve-se os elementos internos.

Servidor *Workflow* Interface – Interface que define os métodos disponibilizados como serviços públicos para os usuários de um processo *Workflow*. Possui os seguintes métodos:

- *Executar* - Recebe como parâmetro um *URL* apontando para um arquivo XML com a definição do processo a ser executado e uma *String* com uma pasta local, na qual devem ser armazenados os arquivos de log de execução das instâncias. Este método só retorna o controle para o solicitante ao final da execução da instância *Workflow*.
- *Disparar* - Recebe os mesmos parâmetros que *Executar*. A diferença é que este método retorna o controle para o solicitante logo após a instanciação e ativação da instância *Workflow*.
- *ExecutarCI* - Possui as mesmas características do método *Executar*. Porém, recebe, além dos parâmetros já informados, três vetores de *strings*, os quais representam os parâmetros passados como contexto inicial, daí o acréscimo das letras *CI* ao nome do método. O primeiro vetor armazena os tipos da variáveis passadas, que podem ser um dos tipos já discutidos na Subseção 2.1.. O segundo armazena os nomes das variáveis passadas. O terceiro vetor deve receber os valores das variáveis que estão sendo passadas como contexto inicial para instância.

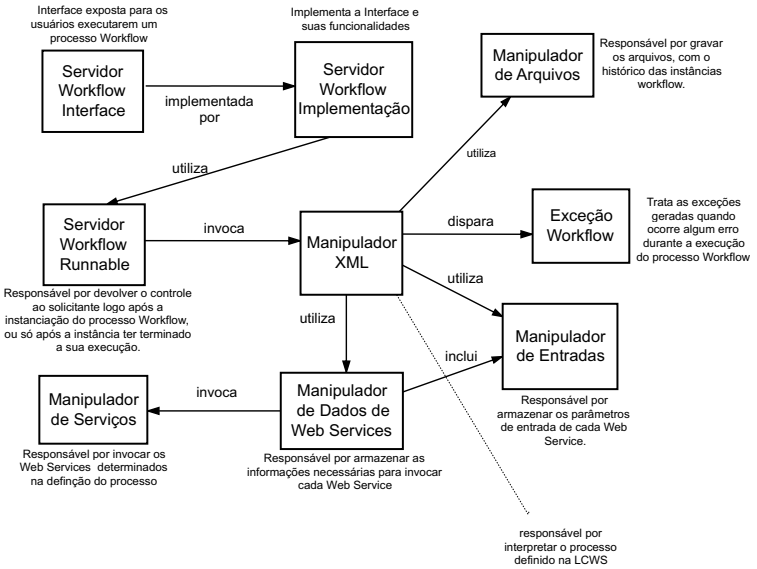

Figura 8. Estrutura interna do AELCWS.

- *DispararCI* - Possui as mesmas características do método *Disparar*. Porém, recebe, além dos parâmetros já informados, três vetores de *strings*, os quais representam os parâmetros passados como contexto inicial, que devem possuir as mesmas informações já relatadas para o método *ExecutarCI*.

Servidor *Workflow* Implementação – Implementa a interface descrita acima, ou seja, os métodos públicos a serem expostos. Manipula as informações recebidas e mapeia os métodos invocados para execução no Servidor *Workflow Runnable*. Esse mapeamento é realizado considerando-se quando o método invocado deve retornar o controle ao solicitante, se logo após a instanciação e ativação da instância, ou somente após o término da execução da mesma.

Servidor *Workflow Runnable* – Responsável por retornar o controle para o solicitante logo após a instanciação e ativação de uma instância do processo *Workflow*, ou após a instanciação, ativação e término da execução da instância. É quem realmente inicia a execução do processo, fazendo uso dos demais elementos que compõe o AELCWS.

Manipulador XML – Pode ser considerado o núcleo do AELCWS, pois é o responsável direto pela interpretação da LCWS. Portanto, é quem interpreta os elementos da linguagem, executando os procedimentos necessários para controlar a execução do processo. Gerencia o controle de fluxo e a manipulação de informações determinados na definição do processo.

Manipulador de Arquivos – Responsável pelo armazenamento dos arquivos XML das intâncias de definições de processos. Esses arquivos XML são os históricos das instâncias *Workflow*.

Exceção Workflow – Quando algum erro ocorre durante a execução da processo, uma exceção apropriada é disparada.

Manipulador de Serviços – Responsável por invocar realmente os serviços, os *Web Services*, durante a execução do processo definido. Conecta-se com o respectivo provedor de cada serviço para realizar a chamada ao serviço necessário para o cumprimento do fluxo do processo *Workflow*.

Manipulador de Dados de *Web Services* – Armazena as informações necessárias para realizar uma conexão com um provedor específico de cada *Web Service*, como, por exemplo, nome do serviço, porta do serviço, tipo de retorno e *name spaces* necessários, dentre outras informações.

Manipulador de Entradas – Armazena apenas informações referentes aos parâmetros de entrada para cada serviço, as quais são capturadas durante a interpretação do fluxo de processo definido na LCWS.

3.4. Sistemas Distribuídos

O AELCWS pode ser considerado como um sistema distribuído, uma vez que este último é um sistema em que componentes de software ou hardware se comunicam e coordenam suas ações apenas pela troca de mensagens [Coulouris et al. 2001].

Os computadores conectados por uma rede podem estar separados no espaço por qualquer distância. Essa característica faz com que surjam alguns desafios a serem tratados no desenvolvimento de sistemas distribuídos. Alguns destes desafios, discutidos em [Coulouris et al. 2001], são atendidos pelo AELCWS e são enumerados a seguir.

1. Heterogeneidade de Componentes, é possível acessar serviços em uma grande variedade de computadores e redes de computadores.
2. É um ambiente Aberto/Livre, do ponto de vista de Sistemas Distribuídos, ou seja, existe a facilidade de adicionar novos recursos compartilhados ou reimplementar recursos antigos, os quais são disponibilizados para diferentes clientes.
3. A Segurança pode ser parcialmente suprida pelo uso de *firewalls* caso os processos sejam executados numa intranet, por exemplo. O aspecto de segurança não foi abordado no desenvolvimento do AELCWS, porém se acredita que esse aspecto pode ser atendido fazendo uso de técnicas de criptografia criadas para este fim.
4. Possui Escalabilidade. Mais *Web Services* em diferentes computadores podem ser disponibilizados sem grandes problemas.
5. Fornece uma abordagem de tolerância a falhas. Existe uma abordagem de tolerância a falhas no contexto dos *Web Services* envolvidos no processo *Workflow*, pois existe a possiblidade de se informar um *Web Service* alternativo para cada *Web Service* principal, o qual é utilizado caso ocorra alguma exceção com o principal. No entanto, não há tratamento de falhas no contexto de uma instância do processo *Workflow* como um todo.
6. Suporta a Concorrência de Componentes. Cada *Web Service* é responsável por garantir sua correta execução num ambiente concorrente.
7. Suporta a transparência. Oculta do usuário e do programador a separação existente entre os *Web Services* utilizados no *Workflow* executado no AELCWS. Assim, um *Workflow* definido na LCWS é percebido como um todo ao invés de ser percebido

como é na realidade, ou seja, como uma composição de *Web Services* independentes.

4. Análise Qualitativa da Proposta

Nesta Seção realiza-se uma análise qualitativa da utilização da LCWS através do AELCWS. Destaca-se as vantagens providas pelo uso dos conceitos e ferramentas apresentados nesta pesquisa. Ilustra-se as vantagens providas pelo uso tanto da tecnologia *Workflow* como da tecnologia *Web Service*.

Essa análise qualitativa está baseada na utilização da linguagem LCWS para realizar a definição de processos, os quais podem ser colocados em funcionamento através do ambiente de execução AELCWS.

Pode-se observar através da avaliação da LCWS, bem como da sua utilização através do AELCWS, que a linguagem para composição de *Web Services* proporciona uma união de vantagens providas pela utilização de tecnologia *Workflow* e tecnologia *Web Service*. Essas vantagens são comentadas a seguir.

A utilização da tecnologia *Workflow* para realizar tal composição fornece as seguintes características:

- **Independência de fluxo –** Decisões de projeto que estejam encapsuladas dentro dos *Web Services* que compõe o processo podem ser revistas e alteradas sem afetar outras partes do processo. Há uma separação entre a lógica do processo e a lógica da aplicação, no caso os *Web Services*. Os *Web Services* não possuem conhecimento da ordem de execução, assim a definição do processo pode ser alterada sem que estes sejam afetados.
- **Independência de Domínio –** A definição de um processo na LCWS, representando um *Workflow*, fornece suporte a qualquer domínio de aplicação, bastando que sejam providenciados os *Web Services* que executem o trabalho específico para o domínio escolhido.
- **Monitoração e Histórico –** Durante a execução de uma instância de uma definição do processo realizada através da LCWS, um histórico para essa instância é criado. É possível rastrear o conjunto de passos executados, o que pode ser utilizado para análise do processo, buscando formas de melhorá-lo. Esse histórico pode ser consultado durante a execução da instância ou após seu término.

A utilização de *Web Services* como atividades de um *Workflow* definido na LCWS fornece as seguintes características:

- Permite que qualquer pessoa descubra dinamicamente os *Web Services*, disponíveis numa intranet, extranet ou Internet, através de um registro de serviços.
- Todo *Web Service* possui sua respectiva descrição, a qual fornece todas as informações necessárias para sua correta utilização.
- Não há necessidade de se possuir conhecimento de como o *Web Service* é implementado realmente, ou a plataforma na qual está sendo executado.

Uma grande flexibilidade é disponibilizada com a utilização da LCWS. A pessoa que realiza a definição do processo precisa saber apenas quais os *Web Services* disponíveis, não necessita saber como são implementados na prática.

É possível alterar a ordem de execução dos *Web Services*, sem afetar suas respectivas implementações. Pode-se, também, alterar a implementação de um *Web Service* qualquer sem afetar a definição do processo do qual este faz parte. Pode-se realizar a troca de um *Web Service* por outro, sem afetar a definição do processo ou a forma como os demais *Web Services* são implementados, sendo necessário apenas que o *Web Service* substituto implemente o mesmo conjunto de funcionalidades.

Outra vantagem que merece destaque é a seguinte: o *Workflow* definido através da LCWS fica disponível como um *Web Service*. Assim, este pode ser utilizado por outras definições de processo como qualquer outro *Web Service*.

A LCWS provê uma grande reutilização de componentes, ou seja, dos *Web Services*. Uma vez adotada a Arquitetura Orientada a Serviços dentro de uma organização, o número de serviços disponíveis na rede corporativa tende a aumentar e consequentemente a reutilização desses serviços também. Dessa maneira, quanto maior for o número de *Web Services* disponíveis, mais rápido se torna o desenvolvimento de processos baseados na composição de serviços, composição esta realizada através da LCWS.

Assim, a utilização da LCWS fornece ao desenvolvimento de processos as seguintes vantagens principais: flexibilidade, reutilização, adaptabilidade a domínios específicos, simples de ser utilizada por ser textual e extensibilidade. Obviamente ela pode ser melhorada com a inserção de novos padrões de estruturas para o controle de fluxo.

5. Conclusão

Nesta Seção são apresentadas as considerações finais do trabalho desenvolvido e também são apresentadas indicações de trabalhos futuros.

5.1. Considerações Finais

Sistemas *Workflow* que fornecem suporte à composição de *Web Services* baseados na arquitetura orientada a serviços é um assunto importante a ser investigado, pois fornece uma grande flexibilidade para definição de processos de negócio, os quais se encontram em constantes modificações. Esses processos são constantemente modificados, pois as necessidades de uma organização devem ser atendidas o mais rápido possível, para que se tornem vantagens competitivas sobre seus concorrentes de mercado. A LCWS permite uma integração tanto de aplicações internas, como de aplicações que transpõem as fronteiras organizacionais.

Este trabalho disponibiliza uma Linguagem Para Composição de *Web Services* (LCWS) e uma arquitetura de Ambiente de Execução para Linguagem Para Composição de *Web Services* (AELCWS). Desta forma, acredita-se ter fornecido uma investigação valiosa sobre o assunto que envolve *Workflow* e *Web Services*.

A LCWS foi analisada no Seção 4., em que foram expostas as principais vantagens da definição de processos fazendo uso desta linguagem. Merece destaque também, que a arquitetura desenvolvida para o ambiente de execução AELCWS foi importante para que se pudesse obter um meio de utilização imediata da LCWS.

5.2. Trabalhos Futuros

Alguns pontos de melhoria que podem ser estudados em trabalhos futuros puderam ser constatados ao final deste trabalho. A seguir, estes pontos são relatados relacionando-os com a LCWS ou com o AELCWS. Note que esse categorização de trabalhos futuros indica apenas se a melhoria se relaciona mais com a LCWS ou com o AELCWS, assim é importante observar que melhorias na LCWS podem requerer alterações no AELCWS e vice-versa.

Melhorias para a LCWS:

- Suporte a um maior número de padrões para controle de fluxo do *Workflow*, além de seqüência, paralelo, condição e laço.
- Desenvolvimento de um editor gráfico para LCWS. De modo que seja possível criar processos de negócio na LCWS de forma transparente.
- Suporte a uma perspectiva organizacional, além da perspectiva de controle de fluxo e de informações. Isto é, fornecer uma estrutura organizacional para o *Workflow*, na qual devem existir responsáveis pela execução das atividades, no caso os *Web Services*.
- Suporte à evolução de processos [Zschornack 2003], ou seja, suporte à alteração da definição do processo em tempo de execução, principalmente ao tratamento das instâncias que estejam executando.

Melhorias para o AELCWS:

- Suporte ao tratamento de eventos, para implementar métodos de espera melhores. De forma, que seja possível parar uma instância *Workflow* que esteja em execução e, após um determinado tempo ou quando uma data específica for alcançada, iniciar novamente a instância para que continue seu processamento do ponto em que foi parada. Com isso, visa-se melhorar o desempenho das instâncias, as quais podem durar dias em execução.
- Inserir métodos de monitoração das instâncias. De maneira, que se possa realizar pesquisas por instâncias que tenham determinadas características, mesmo em tempo de execução.
- Inserir recursos para tratar problemas de segurança e privacidade, tantos dos *Web Services* como do próprio processo *Workflow* definido na LCWS.

Referências

BEA, IBM, Microsoft, AG, S., and Siebel (2003). Specification: Business process execution language for web services version 1.1. Disponível em: <http://www-106.ibm.com/developerworks/library/ws-bpel/>. Acesso em: setembro 2003.

BEA, Intalio, SAP, and Sun (2002). Web service choreography interface 1.0. Disponível em: <http://wwws.sun.com/software/xml/developers/wsci/>. Acesso em: outubro 2003.

Chiu, D., Karlapalem, K., and Li, Q. (2001). E-adome: enacting composite e-services in an advanced workflow environment. *Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International*, pages 311–316.

Coulouris, G., Dollimore, J., and Kindberg, T. (2001). *Distributed Systems Concepts and Design*. Addison Wesley, England, 3 edition.

Gottschalk, K., Graham, S., Kreger, H., and Shell, J. (2002). Introduction to web services architecture. *IBM Systems journal*, 41:170–177.

IBM (2001). Web services flow language (wsfl 1.0). Disponível em: <http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>. Acesso em: julho 2003.

Kiepuszewski, B., Hofstede, A. H. M. t., and Bussler, C. (2000). On structured workflow modelling. *Conference on Advanced Information Systems Engineering, CAiSE, 12*, pages 431–445.

Microsoft (2001). Xlang - web services for business process design. Disponível em: <http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm>. Acesso em: julho 2003.

van der Aalst, W. (2004). Workflow patterns. Disponível em: <http://tmitwww.tm.tue.nl/research/patterns/index.htm>. Acesso em: janeiro 2004.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., and Barros., A. (2002). Workflow patterns. *QUT Technical report. FIT-TR-2002-02, Queensland University of Technology*. Disponível em: <http://tmitwww.tm.tue.nl/research/patterns/report.htm>. Acesso em: outubro 2003.

van der Aalst, W. M. P., Barros, A. P., ter Hofstede, A. H. M., and Kiepuszewski, B. (2000). Advanced workflow patterns. In *Conference on Cooperative Information Systems*, pages 18–29. Disponível em: <http://citeseer.nj.nec.com/vanderaalst00advanced.html>. Acesso em: outubro 2003.

W3Schools (2004). Xml schema tutorial. Disponível em: <http://www.w3schools.com/schema/default.asp>. Acesso em: janeiro 2004.

WfMC (1999). Workflow management coalition - terminoloy & glossary. Specification. WFMC-TC-1011. Disponível em: <http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf>. Acesso em: outubro 2003.

WfMC (2002). Workflow process definition interface - xml process definition language. Specification. WFMC-TC-1025. Disponível em: <http://www.wfmc.org/standards/docs.htm>. Acesso em: setembro 2003.

Wohed, P., van der Aalst, W. M. P., Dumas, M., and ter Hofstede, A. H. M. (2002). Pattern based analysis of bpel4ws. *Technical Report FIT-TR-2002-04, Queensland University of Technology (QUT)*.

Zschornack, F. (2003). Evolução de esquemas de workflow representados em xml. Master's thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre.