2.
Conceitos básicos

Este capítulo aborda diversos conceitos relativos ao domínio desta dissertação baseados na revisão bibliográfica. A seção 2.1 trata de *workflows* e sistemas de gerência de *workflow*. A seção 2.2 discute duas formas de coordenação de *workflow*: Orquestração e Coreografia. Por fim, a seção 2.3 traz um resumo sobre tecnologias móveis mostrando sua utilização como auxílio aos sistemas de gerência de *workflow*.

2.1.
Workflow e sistemas de gerência de *workflow*

Informalmente, *workflow* define um processo que envolve diferentes tarefas e participantes. Neste processo, as tarefas são encadeadas e coordenadas, de forma que documentos, informações ou trabalhos podem ser trocados entre os participantes, seguindo um conjunto de regras que definem o negócio. Uma tarefa é um procedimento que representa uma atividade individual, contribuindo para a realização de um trabalho total que é representado pelo *workflow*. Uma tarefa pode ter dados de entrada e de saídas (documentos, informações, outras tarefas, etc.). Uma tarefa pode ainda ser representada dentro de um *workflow* como uma ação não automatizada, a qual depende da ação do usuário. Uma tarefa pode ser uma unidade básica do *workflow* ou compor um *sub-workflow*.

Um *workflow* pode ser modelado como um grafo direcionado onde os vértices representam as tarefas e as arestas, os fluxos de controle e de dados (Figura 1). O grafo pode ser cíclico e não necessita ser conectado (tarefa E na Figura 1).

Os grafos cíclicos, ou *workflows* com *loops*, não serão abordados nesse trabalho.

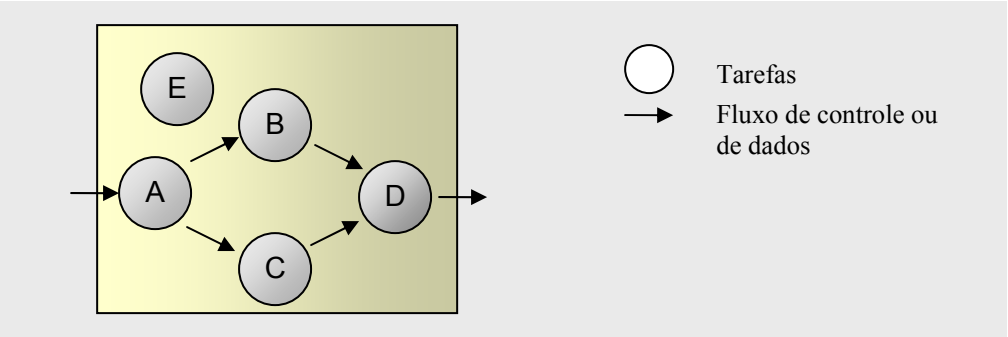

Figura 1 – *Workflow* representado como um grafo

O fluxo de dados diz respeito à troca de informação entre duas tarefas, e o fluxo de controle está associado às regras de dependência entre tarefas. Quando se tem um fluxo de dados entre duas tarefas, normalmente a tarefa sucessora utiliza o dado modificado ou produzido pela tarefa antecessora.

O fluxo de dados entre duas tarefas não necessariamente implica em um fluxo de controle entre as mesmas. Apesar dessa possibilidade, este trabalho assume que se existir um fluxo de dados entre quaisquer duas tarefas, então deve existir um fluxo de controle que represente a relação de dependência entre elas.

Os *workflows* podem ser especificados segundo diferentes perspectivas (van der Aalst et al., 2003) como: a perspectiva de fluxo de controle, que está associada à seqüência de execução; a perspectiva de dados, a qual representa o controle de dados dentro do fluxo de controle; a de recursos, que leva em consideração a estrutura organizacional para prover recursos humanos e de dispositivos para a execução dos processos; e a perspectiva operacional, que é representada por ações elementares executadas pelos processos.

Existem diferentes formas de projetar um *workflow*. Elas dependem da necessidade de cada usuário e do domínio de aplicação do problema em que o *workflow* está inserido. Na fase de análise, ao se projetar um *workflow*, são descobertos os seus requisitos. A depender dos requisitos levantados, determinados modelos de *workflows* devem ser construídos. Como diferentes caminhos para as soluções podem ser tomados, gera-se, às vezes, uma falta de consenso entre os projetistas de modelos.

Na tentativa de criar um consenso na área, existem diversas instituições e pessoas estudando e tentando padronizar modelos de *workflow* conhecidos (WfMC, 2005; van der Aalst et al., 2003; Yu & Buyya, 2005; Russel et al., 2004). A WfMC (*Workflow Management Coalition*) é um dos mais importantes grupos de pesquisa interessados no estudo de padrões e sistemas de gerência de *workflow*.

Atualmente já são conhecidos alguns padrões de *workflow* que abrangem uma grande variedade de problemas. Segundo (van der Aalst et al., 2003), eles podem ser divididos em grupos de características semelhantes como: padrões de controle de fluxo básicos, padrões com ramificações avançadas e sincronização, padrões estruturais, padrões com múltiplas instâncias, padrões baseados em estados e padrões de cancelamento, que vão das mais simples representações as mais complexas formas de se projetar um modelo de *workflow*.

Os padrões de *workflow* abrangidos por este trabalho classificam-se dentro de dois grupos citados: padrões com controle de fluxo básico, que inclui *workflows* com padrões seqüenciais, paralelos e sincronização; e padrões com ramificações avançadas e sincronização, no qual está inserida a junção sincronizada. Na Figura 2 é possível observar um exemplo simples dos quatro padrões de *workflow* abrangidos pelo presente trabalho.

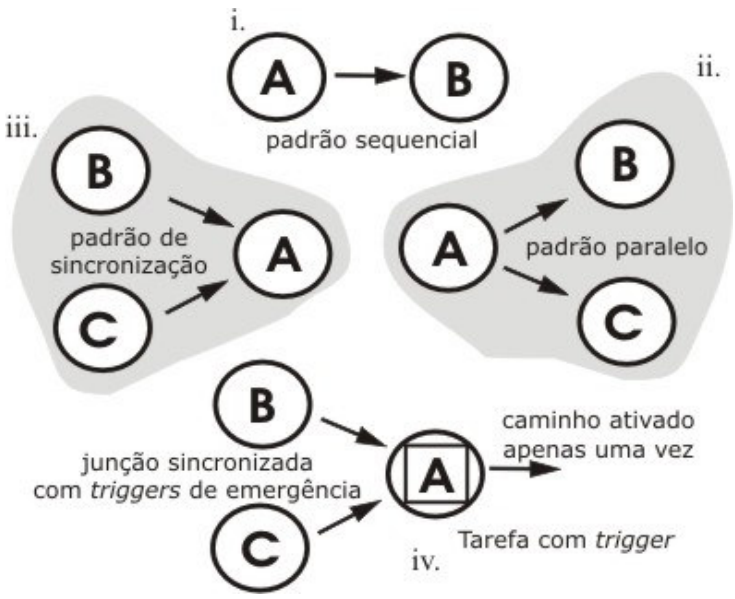

Figura 2 – Representação dos padrões abrangidos nesse trabalho

O padrão seqüencial (Figura 2 (i)), é o mais simples e é facilmente encontrado em todos os modelos de processo de *workflow*. Ele se aplica toda vez que uma tarefa depender unicamente de outra tarefa para continuar sua execução.

No padrão paralelo (Figura 2 (ii)), há um ponto no *workflow* em que há uma divisão, a partir do qual, o fluxo do processo pode ser desencadeado por mais de um caminho. Estes diversos caminhos possuem tarefas que podem ser executadas em paralelo. As tarefas em paralelo podem ser executadas tão logo a tarefa anterior esteja terminada. Estas tarefas são normalmente atividades que não possuem dependências entre si, ou que não disputam simultaneamente os mesmos recursos. É possível também associar condições a cada um das tarefas paralelas (condição ao fluxo que leva a esta tarefa paralela). Neste caso, elas serão executadas após a finalização da tarefa precedente e quando a condição para sua execução for satisfeita. Essa abordagem não é coberta nesse trabalho.

O padrão de sincronização (Figura 2 (iii)) pode ser visto como o inverso do paralelo. Ele está representado quando se têm duas ou mais tarefas paralelas anteriores a uma outra tarefa. As tarefas paralelas convergem para a tarefa subseqüente. É importante notar que nesse padrão, as tarefas paralelas devem ser todas processadas, ou seja, a tarefa seguinte só será executada se todas as anteriores forem concluídas.

A junção sincronizada (Figura 2 (iv)) funciona como uma extensão do padrão de sincronização. Da mesma forma, existe duas ou mais tarefas executando em paralelo convergindo para uma tarefa centralizadora. Entretanto, não necessariamente todas as tarefas precisam ser finalizadas para que a centralizadora continue a sua execução. Ficará a critério do ponto centralizador decidir se, ao receber a conclusão de uma das transições, ainda será necessário esperar pela conclusão de outra transição. Este trabalho não dará suporte a junção sincronizada, mas sim a uma extensão da mesma, chamada aqui de junção sincronizada com *triggers* de emergência.

Na junção sincronizada com *triggers* de emergência a tarefa centralizadora (Figura 2 (iv), tarefa A) é uma tarefa que possui um tempo limite para esperar pela

execução das suas pré-condições. Se passado esse tempo e as pré-condições não estiverem finalizadas, o *trigger* de emergência associado será disparado e a tarefa estará liberada para execução. Esta abordagem é interessante para situações de emergência e será um dos pontos cruciais desse trabalho. No capítulo 5, será discutido mais sobre essa abordagem.

A automatização do projeto, o controle e a execução dos processos são feitas através dos sistemas de gerência de *workflow* (SGWf). Os SGWf auxiliam os usuários na definição e execução dos fluxos de trabalho, com utilização otimizada dos recursos disponíveis. Para a WfMC, um SGWf é um sistema que define, gerencia e executa *workflows*, cuja ordem de execução é dirigida por uma representação informatizada. Existem atualmente diversos sistemas de gerência de *workflow* implementados de diferentes formas, utilizando diferentes tecnologias. Entretanto, estes sistemas possuem características básicas comuns, que servem de base para o projeto de um padrão de interoperabilidade entre tais tipos de sistemas.

O WfMC descreve uma arquitetura com as características básicas que os SGWf possuem e o relacionamento entre suas principais funcionalidades (Figura 3). As funcionalidades são divididas em duas etapas: tempo de construção e tempo de execução. No tempo de construção são feitos o projeto e definição do *workflow* através de ferramentas apropriadas. No tempo de execução, há principalmente a execução do *workflow* através da máquina de *workflow* que pode interagir com outros sistemas ou com o próprio usuário. A definição do processo pode ainda mudar nesse momento, o que dá origem aos *workflows* dinâmicos ou flexíveis.

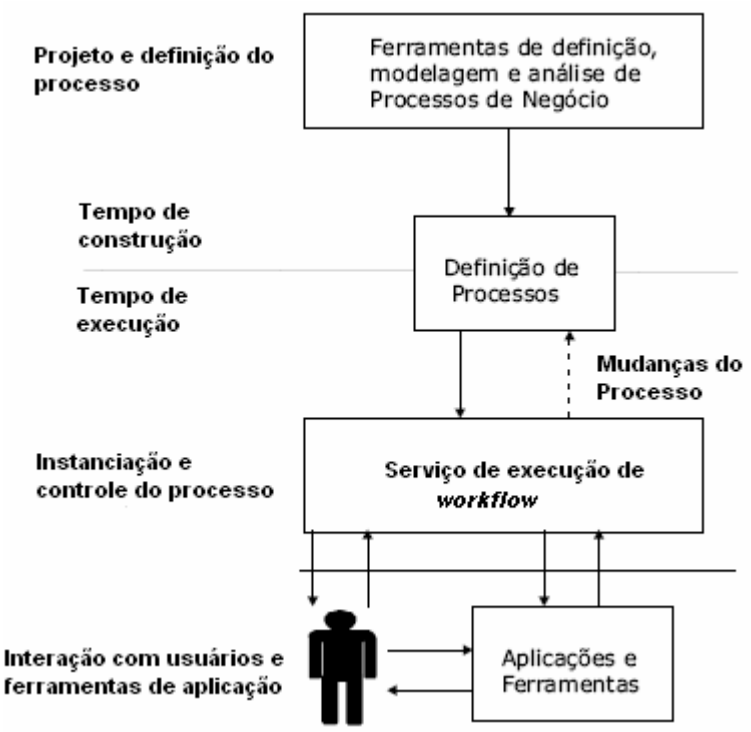

Figura 3 – Características dos sistemas de gerência de *workflow*

A partir desses conceitos básicos, este consórcio definiu alguns padrões, com definições e interfaces relativas aos sistemas de gerência de *workflow*. O modelo de referência da WfMC (Figura 4), traz cinco interfaces básicas que estabelecem um padrão de interoperabilidade entre diferentes sistemas de *workflow*.

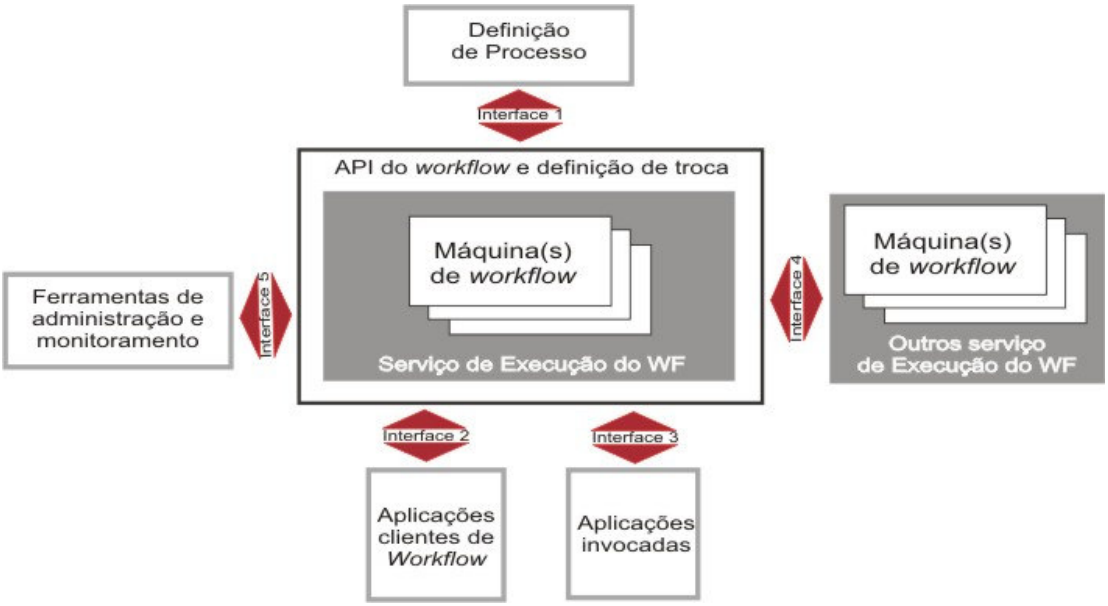

Figura 4 – Modelo de referência do WfMC (Hollingsworth, 1995)

As características gerais de cada uma dessas interfaces são descritas a seguir:

– **Interface 1 – Definição de Processo** - interface padrão entre as ferramentas de modelagem e de definição de processos e as máquinas de execução de *workflow*;

– **Interface 2 – Aplicações Clientes de *Workflow* -** API para as aplicações clientes requisitarem serviços da máquina de *workflow* para controlar o progresso dos processos, tarefas e outros itens de trabalho;

– **Interface 3 – Aplicações Invocadas -** uma API de definição padrão de interface que visa permitir à máquina de *workflow* realizar chamadas a outras aplicações através de agentes de software;

– **Interface 4 – Interoperabilidade de *Workflow* -** definição de um modelo de interoperabilidade entre *workflows* para possibilitar a comunicação com outros sistemas de *workflows*, utilizando-se de padrões para suportar à interoperabilidade;

– **Interface 5 – Ferramentas de administração e monitoramento -** definição de monitoramento e funções de controle.

Essas interfaces cobrem as questões relacionadas aos sistemas de gerência de *workflow*. A padronização de interfaces e formatos de trocas de arquivos garantem uma maior interoperabilidade e confiabilidade entre os sistemas envolvidos,

permitindo que eles possam coexistir ainda que projetado para funcionarem em ambientes heterogêneos.

Há bastante pesquisa relacionada aos SGWf, graças à importância de tais sistemas dentro de uma organização. Diferentes assuntos são tratados na literatura, pelo fato de que muito ainda há para ser feito. Como exemplo, podem ser citados questões relacionadas ao projeto e execução do *workflow*, agendamento das tarefas, tolerância a falhas, movimentação de dados, flexibilização em tempo de execução, *workflows* com suporte à desconexão, entre outros.

2.2. Coordenação de *workflows*

2.2.1. Orquestração e coreografia

Diversas empresas têm escolhido a tecnologia de *Web Services* como solução para desenvolvimento de seus serviços *Web*. Com a explosão do uso da Internet, essa tecnologia tem se mostrado como uma ótima solução para desenvolvimento de serviços para *Web* por usar protocolos padrão da WWW. Desta forma, os *Web Services* permitem uma redução do custo de operação, facilitam o reuso, e consequentemente, melhoram a produtividade, além de funcionar como uma ponte de interação entre diferentes tipos de ambientes. Entretanto, apenas desenvolver produtos não é suficiente para que outras empresas possam reutilizar. É necessário que haja uma descrição de como esses serviços podem ser utilizados individualmente ou em conjunto com outros serviços. Com a linguagem de descrição de *Web Services*, WSDL – *Web Service Description Language* (Christensen & Curbera, 2001), é possível descrever como um *Web Service* pode ser utilizado. WSDL, porém, não mostra como dois ou mais serviços podem ser utilizados em conjunto para atingir um objetivo.

A integração de diversos softwares tem sido cada vez mais utilizada entre as organizações na tentativa de redução de custos, troca de serviços, intercomunicação, reutilização, delegação e divisão de trabalho, entre diversas outras vantagens. Isto tem produzido uma grande variedade de processos de negócios inter-organizacionais, de

forma que a necessidade de tecnologias, teorias ou técnicas para auxiliarem este tipo de ambiente, têm sido bastante estudadas. A coordenação dos processos de negócios, ou *workflows*, exige troca de mensagens, e a tecnologia de *Web Service* tem sido atualmente uma boa abordagem que oferece recursos para suprir esta necessidade. A coordenação de processos de negócios pode ser classificada como orquestração ou coreografia.

A *orquestração*, de uma forma geral, ocorre quando existe um controlador central que coordena as atividades entre vários participantes passivos. Por analogia com uma orquestra musical, o controlador central é o maestro que é o responsável pela organização e execução de todo o processo, ao passo que os participantes são os membros da orquestra, os quais executam as suas tarefas de acordo com a delegação enviada pelo maestro. O processo é controlado sob a perspectiva individual de um participante, o controlador. Este tem o conhecimento de como todas as tarefas individuais, que compõem o processo, funcionam. A orquestração deve ser vista como um processo de negócio executável, em que há uma implementação, um programa que controla todo o fluxo de trabalho, executando as tarefas e trocando mensagens. Os sistemas de gerência de *workflow* configuram um bom exemplo deste tipo de programa.

Existem diversas linguagens, com sintaxe baseada em XML, para descrição de processos de negócios, dentre as quais, BPEL4WS – *Business Process Execution Language for Web Services* – (Andrews et al., 2003), ou BPEL. Esta é uma linguagem robusta que possui seus conceitos fortemente associados à idéia de orquestração, tendendo a ser um padrão de linguagem para este tipo de coordenação de processos (Kavantzas, 2004; Ross-Talbot, 2005a; Ross-Talbot, 2005b).

Coreografia é um modelo que descreve a interação entre dois ou mais serviços para atingir um objetivo global. Diferentemente da orquestração, ela não possui a figura do controlador central. Todos os participantes estão no mesmo nível hierárquico e são tratados igualmente, cada um dos quais com suas devidas responsabilidades dentro da coreografia. Fazendo outra analogia, suponha um grupo de dança que está apresentando uma coreografia musical. Cada um dos integrantes

deve saber quando e o que fazer, para que o objetivo global seja atingido por todos, porém, não é relevante para a coreografia saber como cada um dos participantes faz para executar sua parte no processo. Neste tipo de coordenação existe também troca de mensagens entre os participantes. Entretanto, a principal diferença entre coordenação e orquestração está no fato de que não há uma especificação segundo um único participante, e sim uma descrição global do processo que representa um contrato entre todos os participantes.

A principal linguagem com suporte a coreografia é a WS-CDL – *Web Service Choreography Description Language* – (Kavantzas et al., 2005; Kavantzas & Burdett, 2004; Austin et al., 2004). WS-CDL é uma linguagem de especificação de contratos entre os participantes. Apesar de recente, ainda como candidato recomendado do W3C (W3C, 2006), existe uma forte tendência dela se tornar um padrão para coreografia.

A seguir, será mostrado um exemplo para melhor entendimento desses dois tipos de coordenação de processos.

Suponha um cenário em que há uma compra e venda de produto. A Figura 5 apresenta uma modelagem simplificada deste problema do ponto de vista da orquestração, em que há três participantes: o comprador, o vendedor, e o agente que representa o cartão de crédito. Na Figura 5 pode ser observado que o problema foi separado segundo a perspectiva de cada participante (duas linhas verticais tracejadas) e, para cada um, pode ser gerado seu *workflow* correspondente, que pode ser escrito na linguagem BPEL ou outra qualquer. Cada participante atua como o controlador central da sua parte e, de tempos em tempos, são feitas trocas de mensagens entre os participantes (representadas pelas setas que apontam para ou saem das linhas verticais tracejadas) para requisição de informações. É importante notar ainda que cada *workflow* pode conter etapas, omitidas na figura, que sejam pertinente apenas a ele, não sendo necessário que os outros participantes tenham conhecimento como, por exemplo, acesso à base de dados.

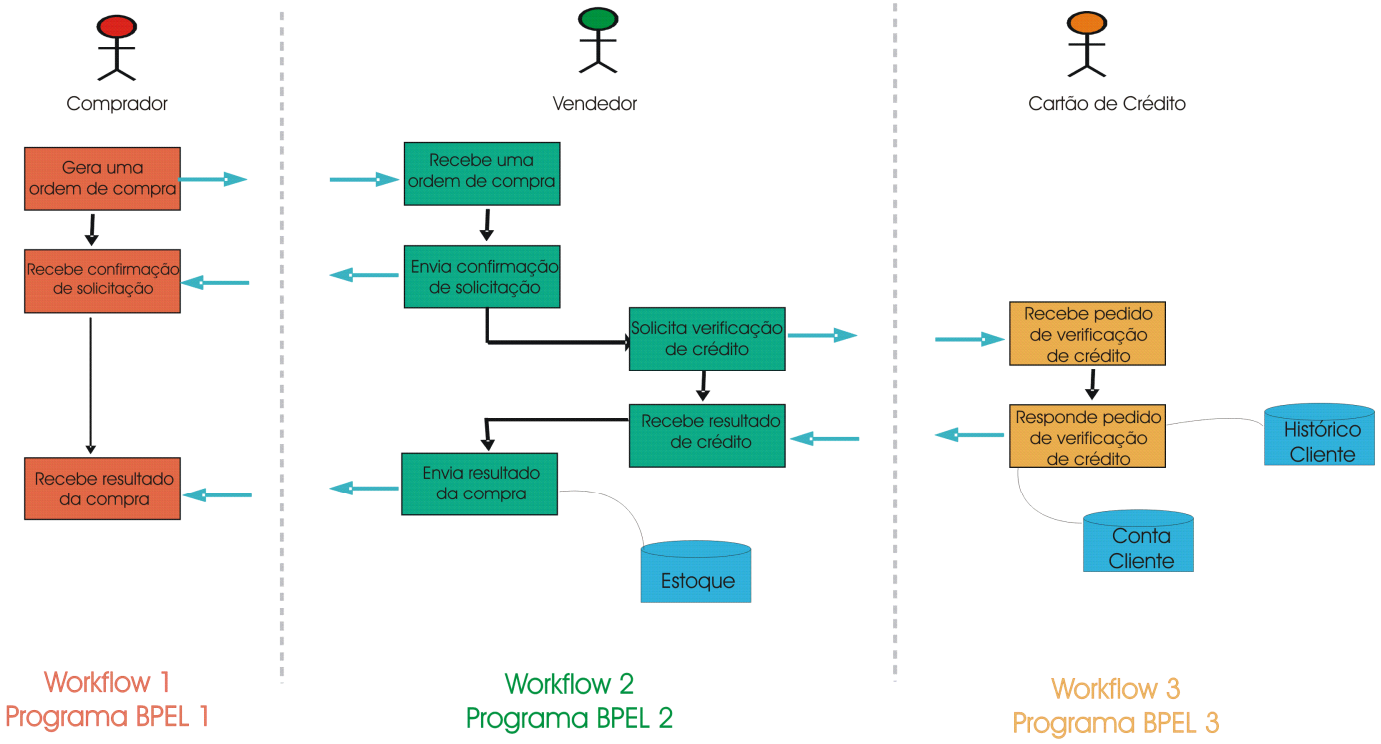

Figura 5 – Orquestração: Compra de produto

A Figura 6 apresenta uma modelagem do mesmo exemplo segundo a coreografia. Neste caso, o objetivo de compra é analisado do ponto de vista global, diferentemente do anterior, que analisava segunda a perspectiva de cada participante. Porém, não há aqui um programa executável, e sim uma especificação de como a venda do produto deve ocorrer como um todo. Essa especificação, que pode ser negociada entre os analistas de negócios das empresas (os participantes) que atuam no processo, funciona como um contrato entre os participantes. Os participantes podem desempenhar diferentes papéis dentro da coreografia. A partir desta especificação, cada participante deve implementar sua parte do processo. Neste momento há uma aproximação da orquestração, porém, não é relevante para a coreografia como cada um constrói sua parte, seja em Java, BPEL, ou outra linguagem.

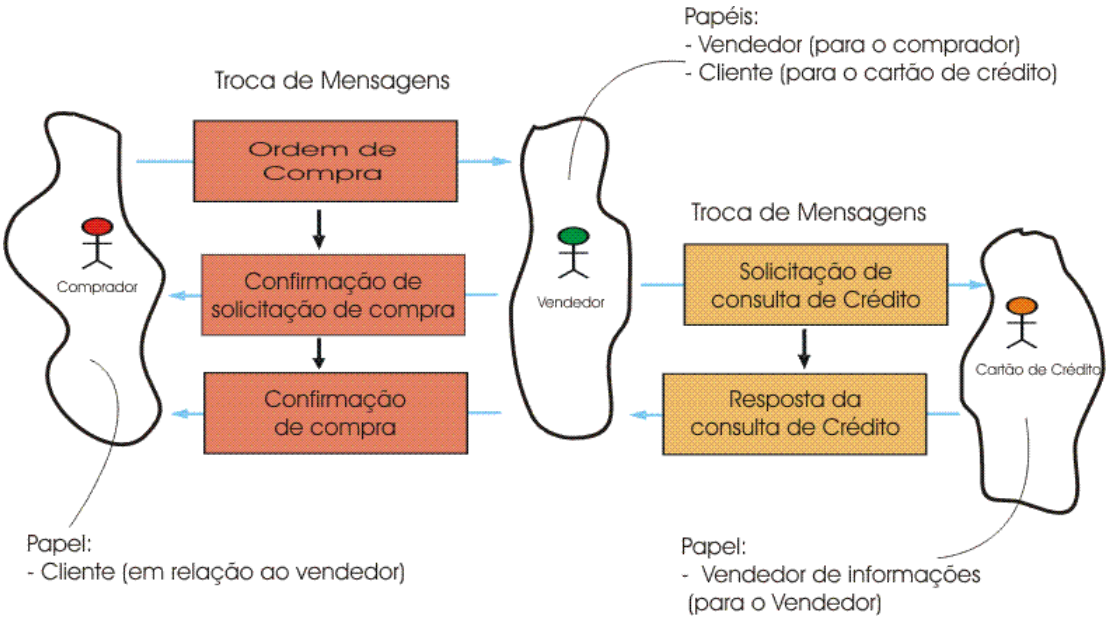

Figura 6 – Coreografia: Compra de produto

A reutilização da coreografia é mais visível que a orquestração. Para exemplificar, suponha que um novo participante queira fazer parte dos dois processos descritos anteriormente. Na orquestração, ele vai ter que descobrir como é feita a troca de mensagens e desenvolver seu *workflow* correspondente. Já na coreografia, basta que ele tenha em mãos o contrato gerado. A partir dessa especificação ele já sabe o papel de cada participante e como são feitas as trocas de mensagens entre os mesmo. Desta forma, ele conhece a forma de comunicação e pode, mais facilmente, integrar-se ao processo.

A Tabela 1 apresenta uma breve comparação entre orquestração e coreografia.

Nas duas subseções seguintes serão mostrados dois resumos sobre as duas linguagens principais para orquestração e coreografia, BPEL4WS e WS-CDL, respectivamente.

	Orquestração	Coreografia
Produto Alvo	Sistema *executável* para cada participante	Uma *especificação* (contrato) entre os participantes
Principal Linguagem	BPEL4WS	WS-CDL
Reutilização	Dá-se ao nível de cada *workflow*	Ao nível do contrato
Modelo de Negócio	Segundo um controlador central	Todos os participantes possuem a mesma hierarquia
Suporte a *Workflow*	Sim	Sim
Tipo de comunicação	Externamente: ponto-a-ponto Internamente: depende da implementação do *workflow*	Externamente: ponto-a-ponto Internamente: não especifica
Composição Recursiva	Sim	Sim

Tabela 1 – Orquestração X coreografia

2.2.2. BPEL4WS

BPEL4WS - *Business Process Execution Language for Web Services*, ou BPEL, é uma linguagem derivada de duas outras linguagens: XLANG (Thatte, 2001) e WSFL (Leymann, 2001). Foi definida em conjunto pelas empresas BEA, IBM, Microsoft, SAP AG e Siebel Systems e atualmente está na versão 1.1. Esta é uma linguagem robusta para construção de processos de negócio, a qual está definida sobre as tecnologias básicas de Web Services como, por exemplo, XML Schema, XPath, WSDL (Web Service Description Language), etc. Tais tecnologias são chamadas de *stateless*, por não permitirem a representação de estados de uma aplicação. Entretanto, as interações dos processos de negócio geralmente representam seqüências de trocas de mensagens ponto-a-ponto com necessidade de manutenção do estado (*stateful*) e especificação da lógica do processo. BPEL agrega valor as estas tecnologias para permitir desenvolvimento de aplicações com estes tipos de necessidades.

Um processo descrito em BPEL define como múltiplos serviços, disponibilizados por diferentes parceiros, podem interagir para atingir um objetivo comum de negócio, seguindo uma ordem e regras previamente definidas. Como visto, ela permite a representação do estado e da lógica necessária para a aplicação. Esta linguagem de processo de negócio possui basicamente todas as funcionalidades de uma linguagem de programação, permitindo a implementação dos mais diversos padrões de *workflow*.

Os processos de negócio podem ser descrito de duas formas: modelo de processo de negócio executável (processo executável), e protocolo de negócio (processo abstrato). O processo executável define o real comportamento de um participante dentro de um processo, enquanto que o processo abstrato é a descrição de processos que descrevem mutuamente as trocas de mensagens entre participantes, sem se preocupar com o comportamento interno. Neste ponto, BPEL aproxima-se dos propósitos da WS-CDL, apresentada na próxima seção.

A Tabela 2 apresenta os principais construtores pertencentes à linguagem.

Construtores padrões	**Definição**
Receive	faz um processo esperar por uma mensagem de solicitação
Reply	permite o processo responder (enviar) as mensagens recebidas através do *receive*
Invoke	permite ao processo invocar uma operação disponibilizada por um parceiro
Invoque	invoca as operações de um único sentido ou *request-response*
Assign	atualiza o valor das variáveis com novos valores
Throw	permite lançar uma exceção de dentro do processo
Wait	espera por um certo período
Empty	permite a inserção de operações sem ação dentro do

	processo
Sequence	agrupa as tarefas que devem ser executadas em seqüência
Switch	escolher uma opção (tem a *default*)
While	repetir uma tarefa até a condição não ser mais satisfeita
Pick	esperar por uma chegada de mensagem ou um *time-out*. Quando isso acontece a atividade dentro desta *tag* é executada
Flow	executar atividades em paralelo
Scope	simular uma classe interna (*inner class*). Permite a definição de atividades aninhadas com suas próprias variáveis, controle de exceções e compensação, etc
Compensate	bloco que é invocado quando se quer compensar algo, deve ser chamado de dentro de um *exception*
Exception	bloco de exceção

Tabela 2 – Principais construtores da linguagem BPEL4WS

2.2.3. WS-CDL

WS-CDL é uma linguagem com sintaxe XML, baseada em WSDL 2.0, que descreve colaborações *peer-to-peer* entre participantes, segundo uma visão global do processo de negócio (Kavantzas et al., 2005). Esta linguagem tem como objetivo definir uma especificação para um modelo global de interações entre os *Web Services* das empresas participantes, permitindo trocas de mensagens, composição de processos, controle de fluxo, independentemente da linguagem de programação e plataforma adotada por cada participante.

Esta linguagem endereça um conjunto de características que buscam descrever formas de como os serviços podem ser utilizados sozinhos ou em conjuntos com outros para alcançar um objetivo comum (Barros et al., 2005), permitindo uma maior integração e reutilização por parte de outros participantes.

WS-CDL é a principal linguagem para descrição de coreografias. Com ela, dois ou mais participantes podem firmar um contrato que especifica como seus serviços devem ser integrados. No documento WS-CDL gerado devem estar especificadas, além das interações, a seqüência e as restrições (regras) sobre as quais as interações devem ocorrer. Cada participante, a partir do WS-CDL gerado, desenvolve sua parte da coreografia utilizando qualquer linguagem de programação ou plataforma desejada. Este documento pode ser reutilizado por outros participantes que desejem fazer uso dos serviços pertencentes a este processo.

A seguir estão listados os principais propósitos da WS-CDL (Kavantzas et al., 2005):

- Definir uma seqüência de interações entre um grupo de *Web Services* que cooperam entre si;
- Promover um comum entendimento entre os participantes;
- Validar automaticamente as coreografias;
- Garantir interoperabilidade;
- Fazer um forte controle de exceção;
- Gerar esboço de código;
- Reuso: a mesma especificação de coreografia poderá ser usada por todos os participantes em diferentes contextos, e diferentes softwares;
- Composição recursiva: coreografias podem ser criadas a partir de outras já existentes.

Estrutura do documento WS-CDL

Nesta seção será mostrada, em alto nível, uma visão conceitual da estrutura do documento WS-CDL (Figura 7). Nesta figura estão representadas as principais estruturas da linguagem WS-CDL.

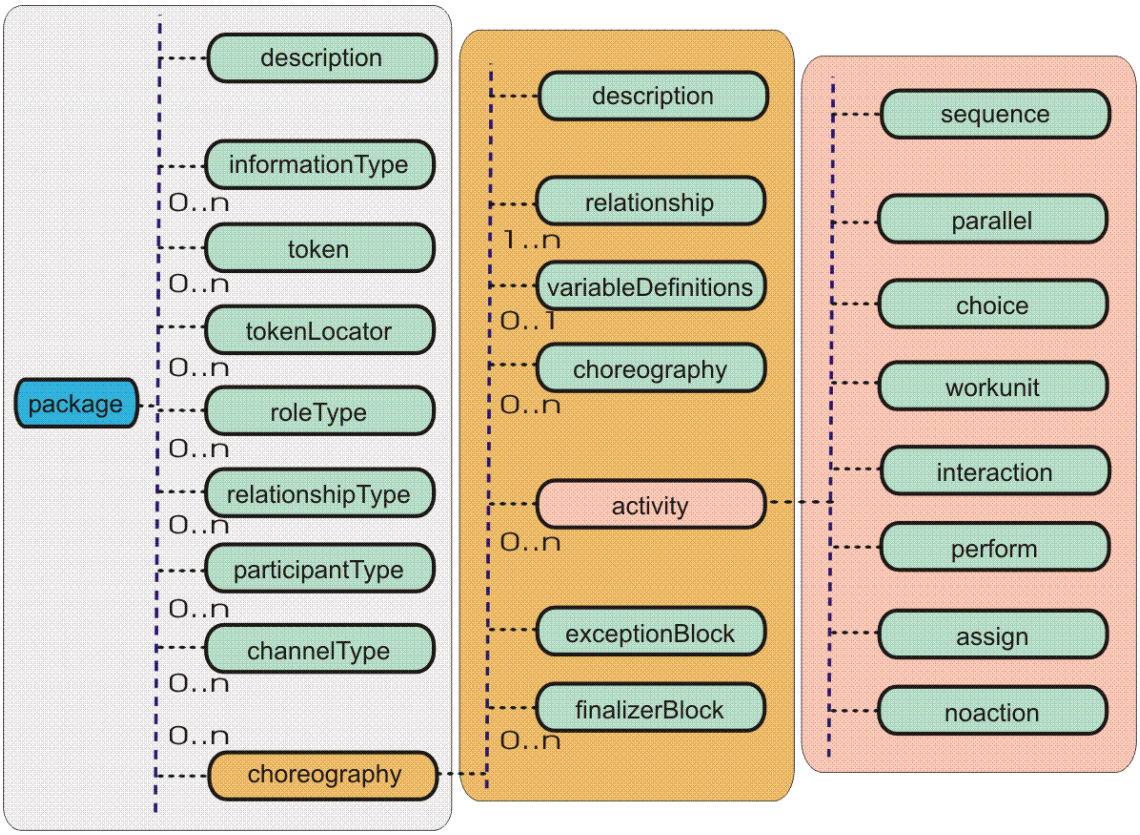

Figura 7 – Pacote WS-CDL

1. package: representa a estrutura do pacote principal da coreografia. É a *tag* raiz que engloba todas as outras definições. As especificações 0..n, 1..n, 0..1 indicam a quantidade de atributos do tipo em questão que podem existir dentro da coreografia (n representa qualquer quantidade). Por exemplo, o *package* pode conter qualquer número de *tokens* e a *choreography* contém zero ou uma *variableDefinitions*.

1.1. description: aqui é possível definir atributos de alto nível que trazem informações sobre a criação da coreografia, tais como nome, autor e versão.

1.2. informationType: descreve o tipo de informação que a coreografia contém. Pode ser o tipo de uma variável ou o tipo ao qual um token referencia.

1.3. token: é uma referência para uma parte da informação contida em uma variável ou para uma mensagem que será usada posteriormente pela coreografia. Token possui um atributo informationType que define o seu tipo de dado.

1.4. tokenLocator: é através deste parâmetro que se tem uma forma de acessar os tokens e posteriormente informações contidas neles.

1.5. roleType: este parâmetro representa o papel desempenhado pelo participante da coreografia. Um participante pode integrar uma coreografia desempenhando diversos papéis (ex.: vendedor, comprador).

1.6. relationshipType: define os relacionamentos entre os roleTypes. O papel vendedor possui o relacionamento de venda com o comprador.

1.7. participantType: define os participantes da coreografia. Esta tag agrupa os comportamentos que podem ser representados por uma empresa ou organização que participa da coreografia.

1.8. channelType: define o canal de comunicação entre os participantes, especificando onde e como as informações são trocadas.

1.9. choreography: Neste pacote são definidas a regras para as requisições de trocas de mensagens. Ela possui diversos atributos que serão descritos a seguir.

1.9.1. description: contém as descrições em alto nível do elemento *choreography*. Assim como no *package*, a descrição pode agregar atributos de autoria como nome, autor e versão.

1.9.2. relationship: define os relacionamentos que uma coreografia pode participar.

1.9.3. variableDefinition: contém a definições de variáveis pertencentes a coreografia. As variáveis contêm valores que podem ser preenchidos através de resultados de ações dentro da coreografia.

1.9.4. choreography: uma coreografia pode conter outras coreografias em sua definição. Estas coreografias definidas aqui são as locais, enquanto que as definidas no package são denominadas de globais.

1.9.5. exceptionBlock: bloco de exceção definido dentro da coreografia. É um bloco que contém ações que podem ser executadas caso ocorra alguma exceção no processo. Fazendo um paralelo com a linguagem java, ele funciona da mesma forma que o catch do bloco try.

1.9.6. finalizerBlock: são ações que, caso sejam definidas, devem ser executadas depois que a execução da instância da coreografia é completada. Na linguagem java, seria o finally. Neste bloco podem ser confirmadas ou desfeitas as ações da coreografia, funcionando, portanto, como um "commit" ou "rollback" de um banco de dados.

1.9.7. activity: descreve as ações que são executadas dentro da coreografia. Este é um elemento importante da coreografia no qual é possível definir, entre outras coisas, padrões de *workflows* como seqüencial, paralelo, escolha e laços.

1.9.7.1. sequence: a estrutura *sequence* agrupa uma ou mais tarefas que devem ser executadas em seqüência, na ordem como foram especificadas.

1.9.7.2. parallel: todas as tarefas desse grupo podem ser executados em paralelo. São casos de tarefas que não têm dependência entre si, ou que não disputam por recursos.

1.9.7.3. choice: escolhe uma tarefa dentre o grupo de uma ou mais tarefas definidas nessa estrutura.

1.9.7.4. workunit: este é um poderoso construtor da linguagem. Com ele é possível definir comandos condicionais "if" ou laços de repetição. Dentro da sua estrutura é possível definir outras atividades.

1.9.7.5. interaction: este é o bloco básico da coreografia. Com ele é possível efetuar a troca de informações entre os participantes.

1.9.7.6. perform: está atividade é a responsável por executar outras coreografias. Estas coreografias instanciadas podem ser as coreografias importadas ou que estejam no mesmo pacote. A atividade *perform* tem um mecanismo que permite o preenchimento das variáveis da coreografia instanciada, permitindo assim a transferência de informações entre as coreografias.

1.9.7.7. assign: com esta estrutura é possível atribuir valores a variáveis. Um valor pode ser transferido de uma variável, chamada de fonte de dados, para outra, que é chamada de alvo.

1.9.7.8. noaction: como o próprio nome induz, esta é uma atividade sem ação. Além da noaction, existe a silentAction. Está última, por sua vez, possui uma ação, porém sem impacto sobre a coreografia.

2.3. Tecnologias móveis e *workflows*

Uma tecnologia móvel pode ser entendida como qualquer tecnologia que possa ser utilizada por usuários em movimento. Este tipo de tecnologia, inicialmente, era apenas um atrativo pela facilidade adicional que oferece. Hoje, porém, ela tem se tornado uma necessidade face às necessidades da sociedade moderna. Como exemplos de tecnologias móveis, podem ser citados celulares, *notebooks*, computadores de mão (*handhelds*), e, indo um pouco além, a junção da telefonia móvel e dos computadores de mão, os chamados *smartphones*.

Muitos desses exemplos de tecnologias móveis são classificadas como PDAs (*Personal Digital Assistent*) ou PIM (*Personal Information Managers*), que é o foco deste trabalho. Os PDAs podem ser os *smartphones* ou os *handhelds*. Suas principais características ficam evidenciadas quando comparadas com um computador pessoal comum:

- tamanho reduzido;
- pouca memória;
- baixo processamento;
- tempo de bateria limitada;
- interface com usuários limitada.

Além dessas, são características marcantes destes dispositivos:

- *touch screen*: o usuário pode manuseá-lo com toques diretamente na tela do dispositivo;
- mobilidade.

Os PDAs começaram a aparecer na década de 70 (Koblentz, 2005) e passaram por grandes evoluções até o presente momento. Sem precisar ir muito longe, no ano

de 2001, foi lançado o Palm m100 (fabricado pela Palm, Inc), um dispositivo atraente para a época, e que possuía apenas 2Mb de memória e tela monocromática. Os usuários desses dispositivos possivelmente ficaram bastante satisfeitos com a evolução de tais aparelhos. Atualmente, com os cartões de expansão, já é possível se trabalhar na casa dos gigabytes. Além disso, os PDAs, ainda que não tanto quanto desejado, estão mais robustos, podem fazer ligações, conectar-se à Internet, usufruindo de muitos recursos da rede, dentre muitas outras vantagens.

Com o avanço desses dispositivos, era de se esperar que muitas soluções de negócios fossem tomadas levando em conta a sua utilização. Eles são vantajosos por permitirem a mobilidade do usuário, a integração em tempo real, onde quer que esteja. Desta forma, dado que uma aplicação tenha característica de mobilidade, seja de força de vendas, controle de negócio com atuação em campo, é praticamente provável que ela conterá soluções baseadas na utilização de PDAs. Esses aparelhos já são encontrados em muitos lugares como hospitais, fazendas (e.g. controle de gado), restaurantes, empresas, etc., além de uso pessoal.

Seguindo essa tendência, os sistemas de gerência de *workflow* passaram a fazer uso das vantagens desses dispositivos. Em 1995 surgiram as primeiras investigações a respeito do tema *workflow* e ambientes com desconexão através do trabalho Exótica/FMDC (Alonso et al., 1995). Este trabalho não está relacionado especificamente com utilização de PDAs, porém, discute questões relativas à desconexão apresentando uma arquitetura distribuída, na qual há um servidor central e clientes desconectados participando do *workflow*. Desta forma, contribui para os *workflows* com dispositivos móveis, os quais também endereçam problemas de desconexão.

Após este trabalho, diversos outros têm discutido o tema, alguns dos quais foram apresentados na Seção 1.4.

Existem diversos cenários de *workflows* que possuem partes que devem ser executadas em campo. Como exemplo, pode ser citado um representante de uma empresa de bebidas, que trabalha na rua junto dos seus clientes. Decisões de estoque e vendas podem (e às vezes precisam) ser tomadas em campo. Outro exemplo comum

é a contenção de desastres naturais. Os participantes do processo vão a campo para realizar as suas tarefas. Resultados parciais podem ser utilizados para tomadas de decisões futuras, e, em se tratando de “desastres”, a integração em tempo real, onde quer que esteja, é importantíssima para conseguir um bom resultado.

O Instituto Brasileiro de Geografia e Estatística, IBGE, está planejando realizar o censo de 2007 com a utilização de milhares de PDAs (IBGE, 2006; Mundo Geo, 2006). A solução de negócio pode ser modelada segundo o *workflow* com suporte à desconexão com utilização de PDAs. Desta forma, tomadas de decisões poderiam ser efetuadas em tempo real, no momento em que o recenseador fizer a coleta dos dados, minimizando, dentre outros, custo de deslocamento. O custo de deslocamento é minimizado, por exemplo, no momento em que um recenseador, atendendo a uma mudança de estratégia, se deslocar para uma área vizinha sem ter que retornar para a base.