Coordenação e Composição de Web Services

Tecnologias de Middleware

DI - FCUL - 2006

João Nogueira

Coordenação de Web Services

Coordenação de Web Services

Motivação

- Em aplicações reais, as interacções são tipicamente mais complexas que invocações simples e independentes a serviços
- A utilização de um determinado serviço envolve, tipicamente, a execução de sequências de operações numa determinada ordem.
  - Estas podem, por vezes, envolver mais que um Web Service!
- A passagem de operações simples e independentes para sequências de operações onde a ordem é crucial tem implicações:
  - Do ponto de vista interno - implementação
  - Do ponto de vista externo - interacção

Coordenação de Web Services

Motivação - Implicações da Coordenação

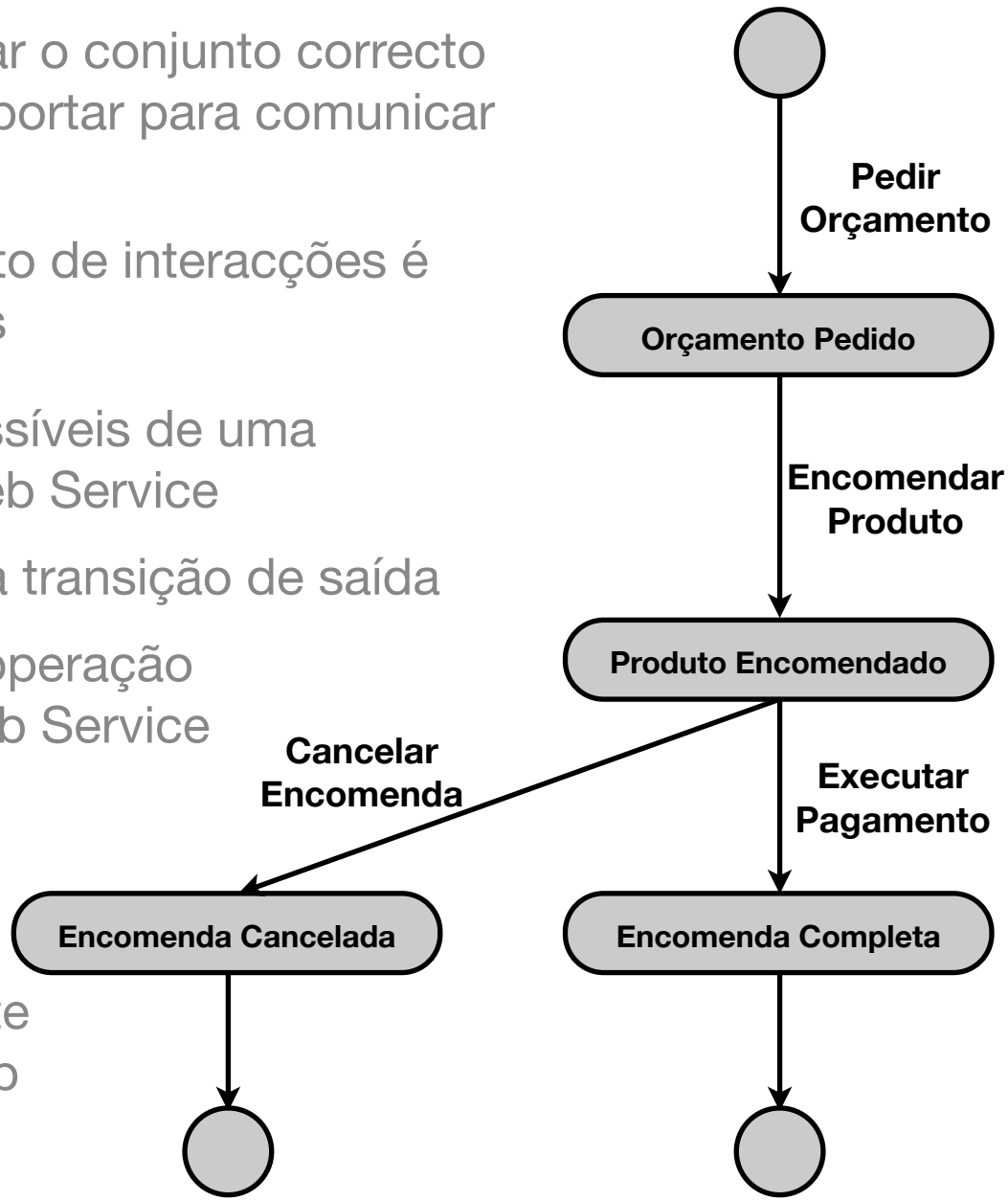
- Do ponto de vista interno, os serviços devem ser capazes de:
  - Executar procedimentos complexos que incluam as operações necessárias pela ordem correcta
  - Manter informação sobre o contexto dos procedimentos que se estendem por várias operações
    - e.g. identificador da compra deve passado de uma operação para a próxima

- Do ponto de vista externo, são impostas restrições aos serviços a nível de:
  - Que operações invocar
  - Quando as invocar
  - A ordem de invocação

Coordenação de Web Services

Modelação da Interacção entre um Cliente e um Web Service

- Como pode um Web Service especificar o conjunto correcto de interacções que um cliente deve suportar para comunicar com ele?
  - Uma forma de modelar esse conjunto de interacções é através de uma máquina de estados
  - Esta descreve todos os estados possíveis de uma conversa correcta entre cliente e Web Service
  - Cada estado pode ter mais que uma transição de saída
  - Uma transição corresponde a uma operação disponibilizada pela interface do Web Service

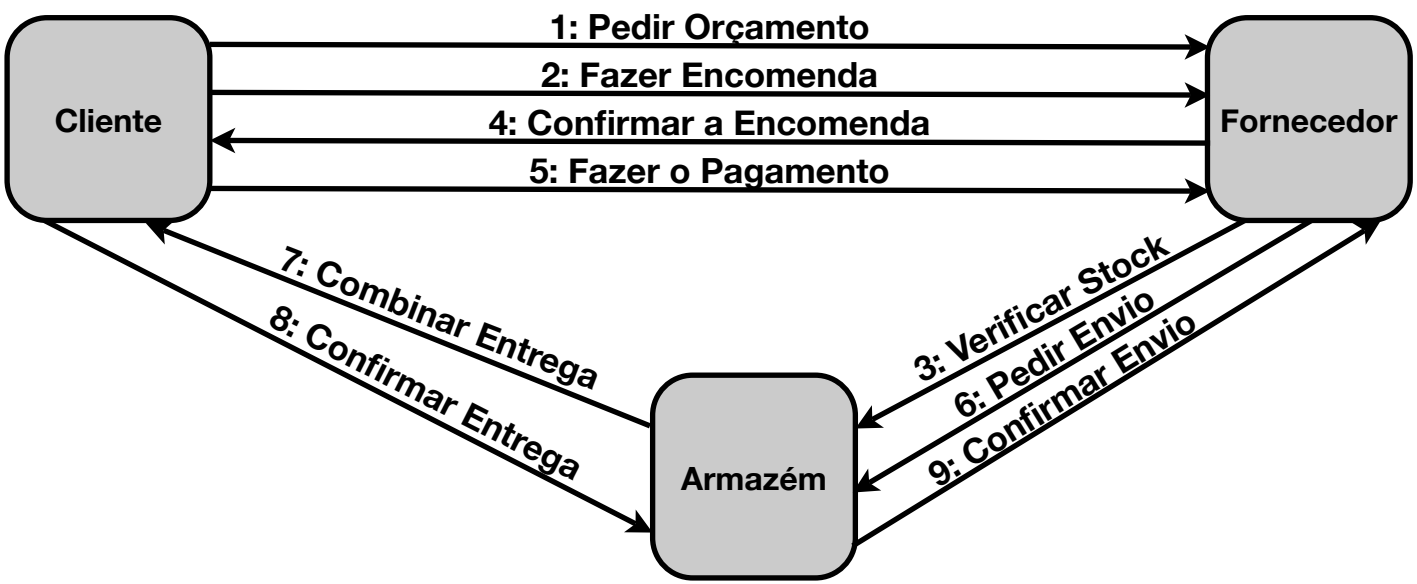


- Usando um diagrama deste tipo e a especificação do Web Service (e.g. WSDL), é possível definir correctamente a interacção entre um cliente e um Web Service

Coordenação de Web Services

Modelação da Interacção entre vários Web Services

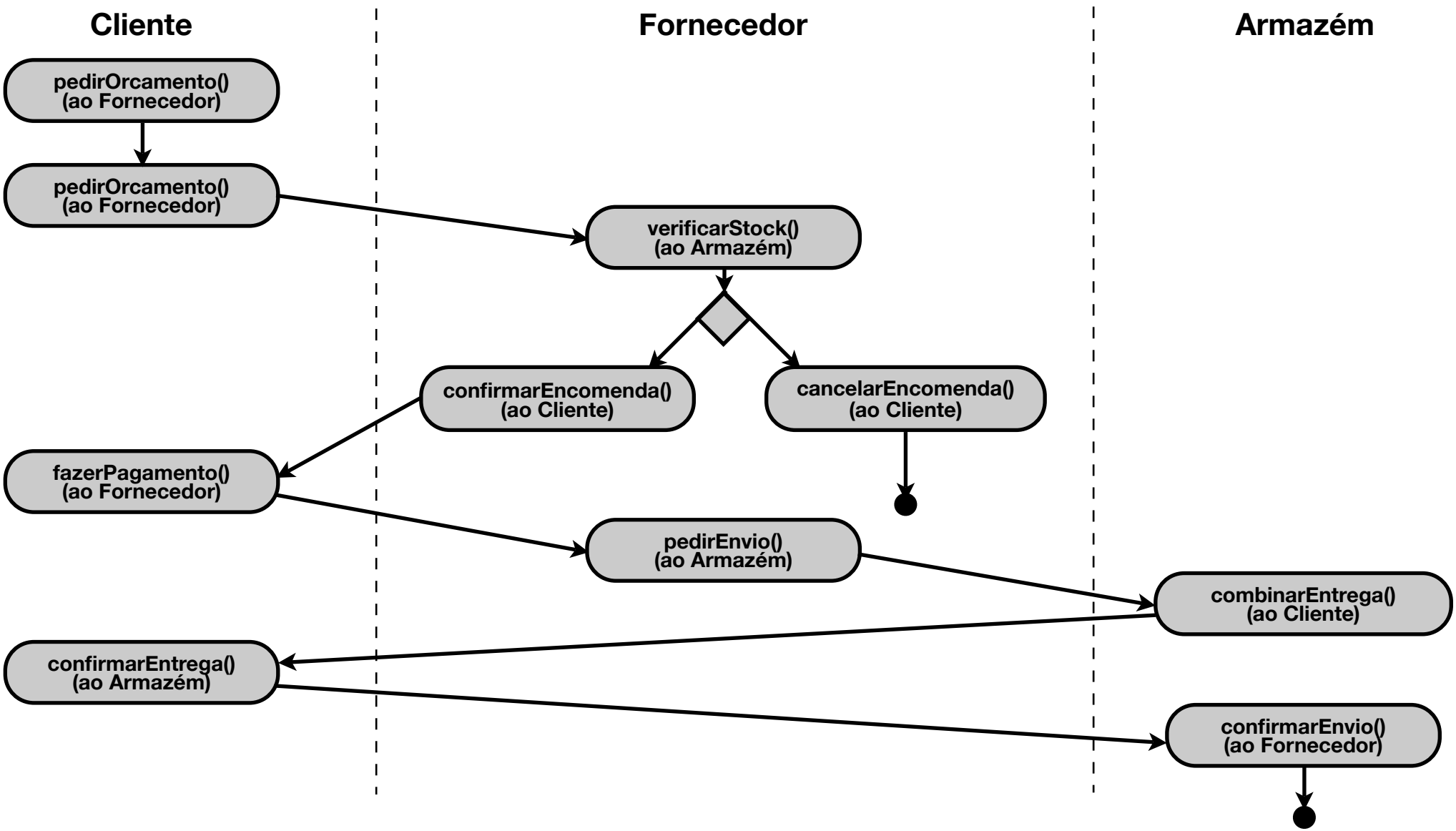
- A solução anterior apenas resolve o problema de oferecer ao cliente uma especificação das sequências de operações que são correctas
  - No caso da interacção entre vários Web Services, cada um impõe as suas restrições, complicando o problema
  - Quem invoca operações pode ser também um Web Service que espera que as operações sejam executadas seguindo uma determinada ordem



Coordenação de Web Services

Modelação da Interacção entre vários Web Services (2)

- Outra forma de modelação é através de diagramas de actividades:



Coordenação de Web Services

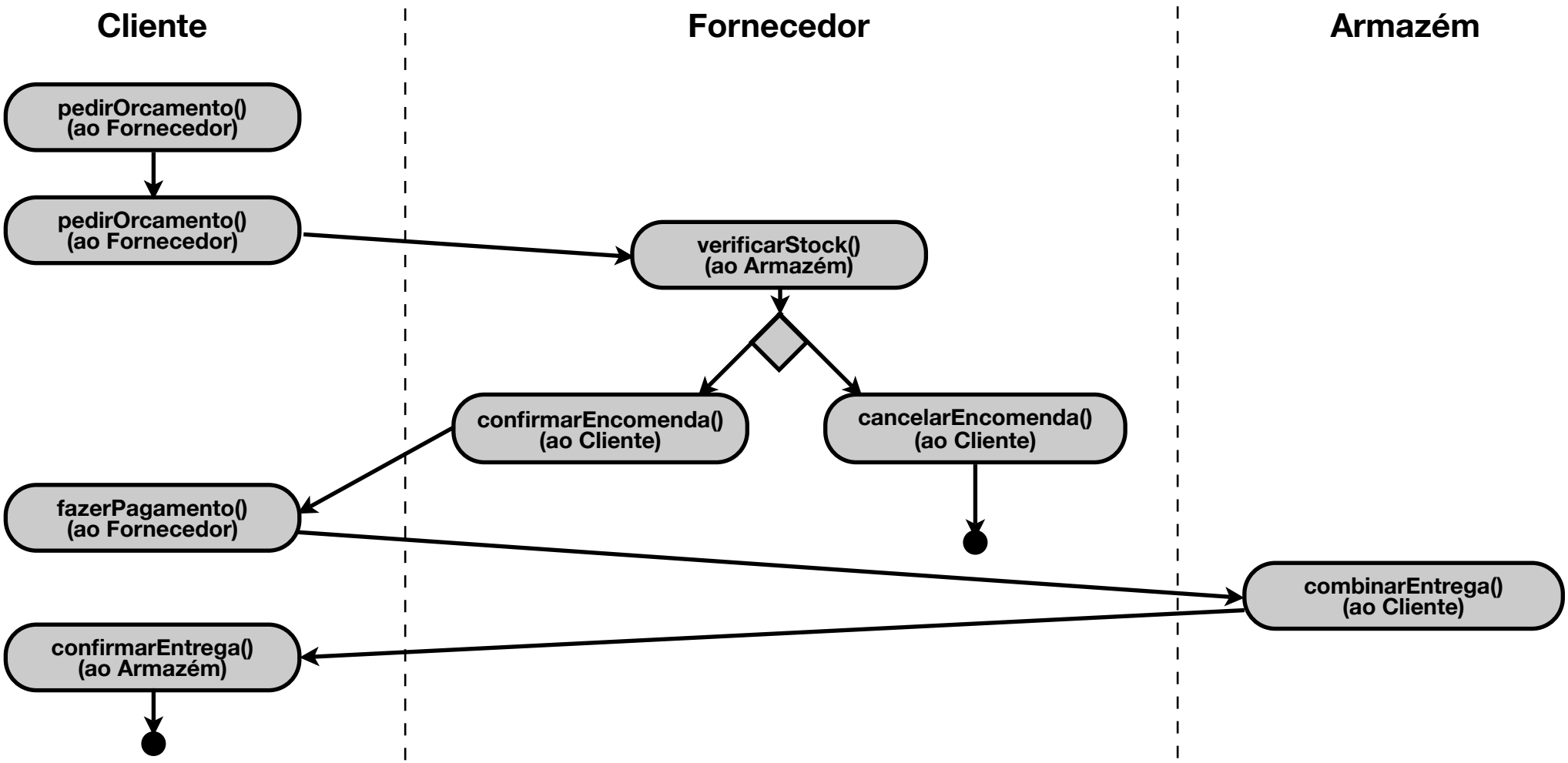
Interfaces dos Servicos e os Protocolos de Coordenação

- Há várias semelhanças entre as interfaces dos Web Services e os protocolos de coordenação:
  - Descrevem como interagir com os Web Services
  - Têm como objectivo facilitar a descoberta dos Web Services em tempo de programação (design-time discovery) e a ligação em tempo de execução (run-time binding)
  - Podem ser publicados em repositórios UDDI como *tModels*. Os papeis dos vários Web Services no protocolo podem ser descritos através de *bindingTemplates* associados a esse *tModel*
  - As descrições do protocolo podem ser usadas para gerar *esqueletos* que suportem a implementação de um serviço que tome um determinado papel
  - Escondem os detalhes de implementação, de forma a proteger a lógica de negócio da empresa e tornar a interacção mais robusta em relação a alterações

Coordenação de Web Services

Interfaces dos Servicos e os Protocolos de Coordenação (2)

- As capacidades de esconder os detalhes de implementação permite também a criação de ***vistas do protocolo*** específicas para cada interveniente que toma um determinado papel.



Coordenação de Web Services

Classificação dos Protocolos de Coordenação

- Protocolos Verticais:
  - Específicos a uma determinada área de negócio
  - Especificam, em grande detalhe, como executar transacções concretas (relativas a uma determinada área de negócio), os documentos envolvidos e seu formato e a semântica do conteúdo desses documentos e das operações que os enviam/recebem
  - O protocolo anterior (compra) é um exemplo de um protocolo vertical
  - São módulos específicos de um determinado domínio de negócio
  - Não se focam nos problemas de baixo nível (e.g. como são trocadas as mensagens, encapsulamento, segurança no canal de comunicação, encaminhamento), mas sim nos aspectos semânticos (e.g. garantir o fluxo correcto das interacções).
    - Baseiam-se, então, tipicamente, na infra-estrutura oferecida pelos protocolos horizontais

Coordenação de Web Services

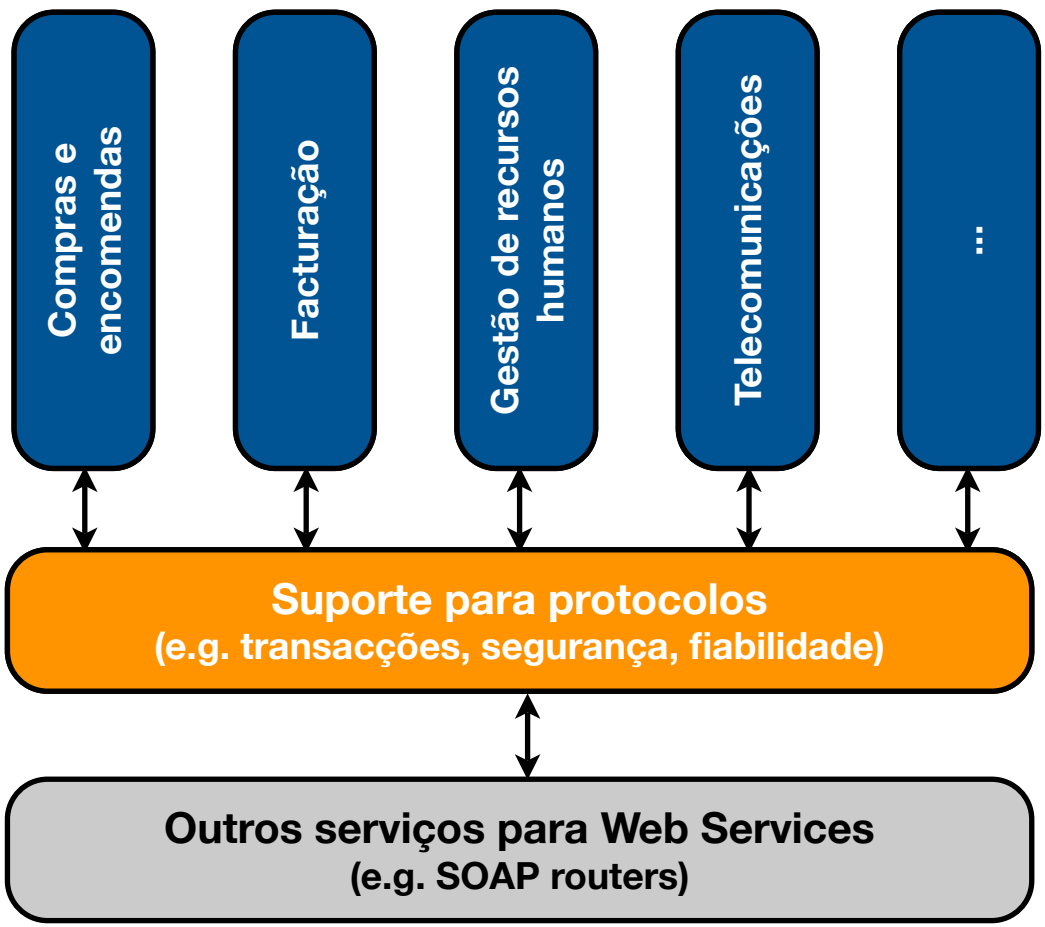
Classificação dos Protocolos de Coordenação (2)

- Protocolos Horizontais:
  - Definem uma infra-estrutura independente da aplicação e da área de negócio
  - Oferecem, às abstracções de nível superior, suporte para interacções com garantias adicionais, quando necessário (e.g. fiabilidade, segurança, operações transaccionais, gestão)
  - Estas propriedades desejadas são, em muitos casos, semelhantes às oferecidas pelo *Middleware* convencional (e.g. RPC, Transaction Monitors)
    - Contudo, no contexto dos Web Services, a maior complexidade do sistema (e.g. a extensão das aplicações por várias empresas, falta de confiança entre os intervenientes) faz com que alguns pressupostos feitos pelo *Middleware* convencional se tornem inválidos:
      - E.g. chamadas bloqueantes a RPC's ou *lock* de recursos durante toda uma transacção não são viáveis no contexto dos Web Services

Coordenação de Web Services

Classificação dos Protocolos de Coordenação (3)

- Os protocolos verticais são muitas vezes combinados com protocolos horizontais, executados pelo *Middleware* que lhe oferecem as propriedades necessárias à sua execução



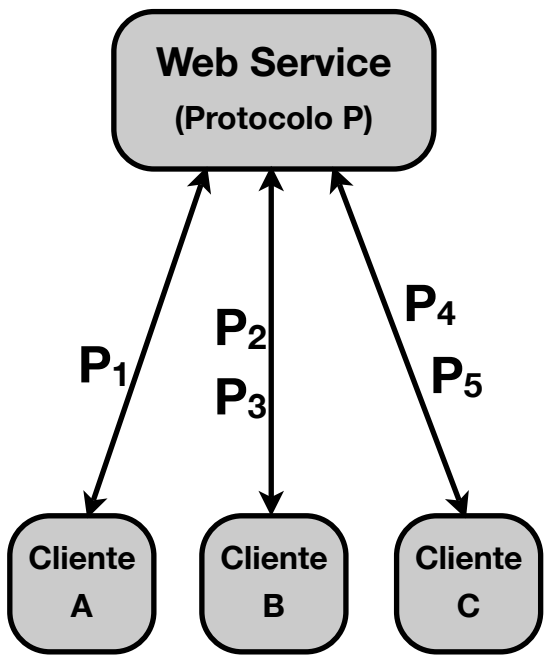
Protocolos Verticais

Protocolos Horizontais

Coordenação de Web Services

Infra-estrutura para Protocolos de Coordenação (1)

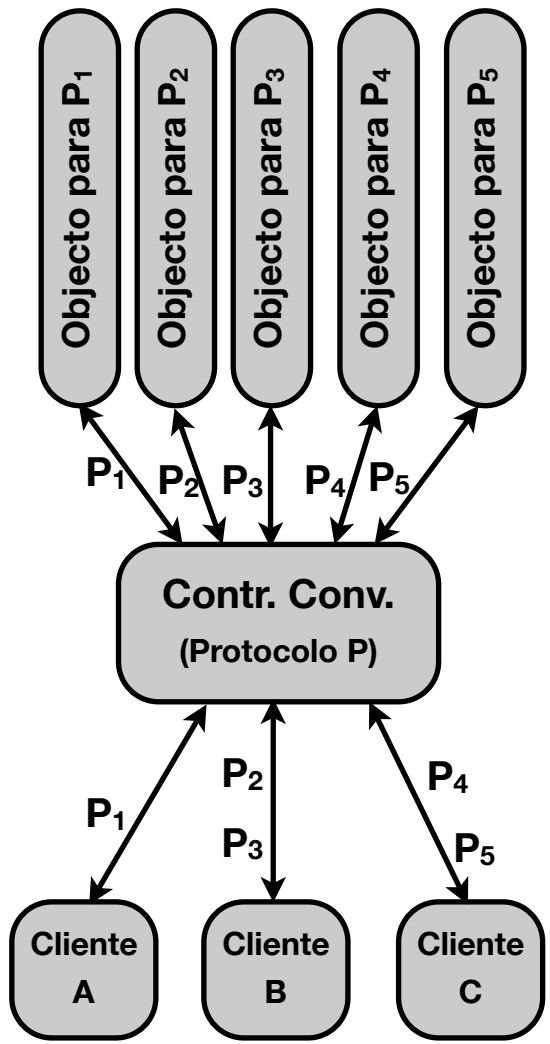
- Controladores de Conversa (*Conversation Controllers*)
  - São facilitadores das interacções entre Web Services
  - As suas funções principais são:
    - **Encaminhar** as várias mensagens para os componentes correctos, de acordo com o protocolo (cada conversa tem estado)
    - **Verificar** se as interacções estão de acordo com a especificação do protocolo e evitar violações desta
  - Pode estar concretizado em cada Web Service



Coordenação de Web Services

Infra-estrutura para Protocolos de Coordenação (1)

- Controladores de Conversa (*Conversation Controllers*)
  - São facilitadores das interacções entre Web Services
  - As suas funções principais são:
    - **Encaminhar** as várias mensagens para os componentes correctos, de acordo com o protocolo (cada conversa tem estado)
    - **Verificar** se as interacções estão de acordo com a especificação do protocolo e evitar violações desta
  - Pode estar concretizado em cada Web Service



Coordenação de Web Services

Infra-estrutura para Protocolos de Coordenação (2)

- Para além dos Controladores de Conversa genéricos, o *Middleware* pode incluir módulos que implementem protocolos de coordenação específicos: *Protocol Handlers*
  - Podem ser criados para qualquer tipo de protocolo, mas é para os horizontais que são mais usados:
    - É de grande utilidade a criação de *protocol handlers* que concretizem protocolos genéricos que ofereçam determinadas garantias (e.g. fiabilidade, transacções atómicas).
    - A sua implementação deve ser genérica para que possam ser aplicados como parte da infra-estrutura de *Middleware* dos Web Services
  - Os protocolos verticais incluem, tipicamente, uma lógica proprietária com detalhes de implementação muito específicos e, por vezes, confidenciais:
    - Não são, normalmente, bons candidatos para este tipo de concretização modular, mas sim para concretizações dentro dos próprios Web Services

Coordenação de Web Services

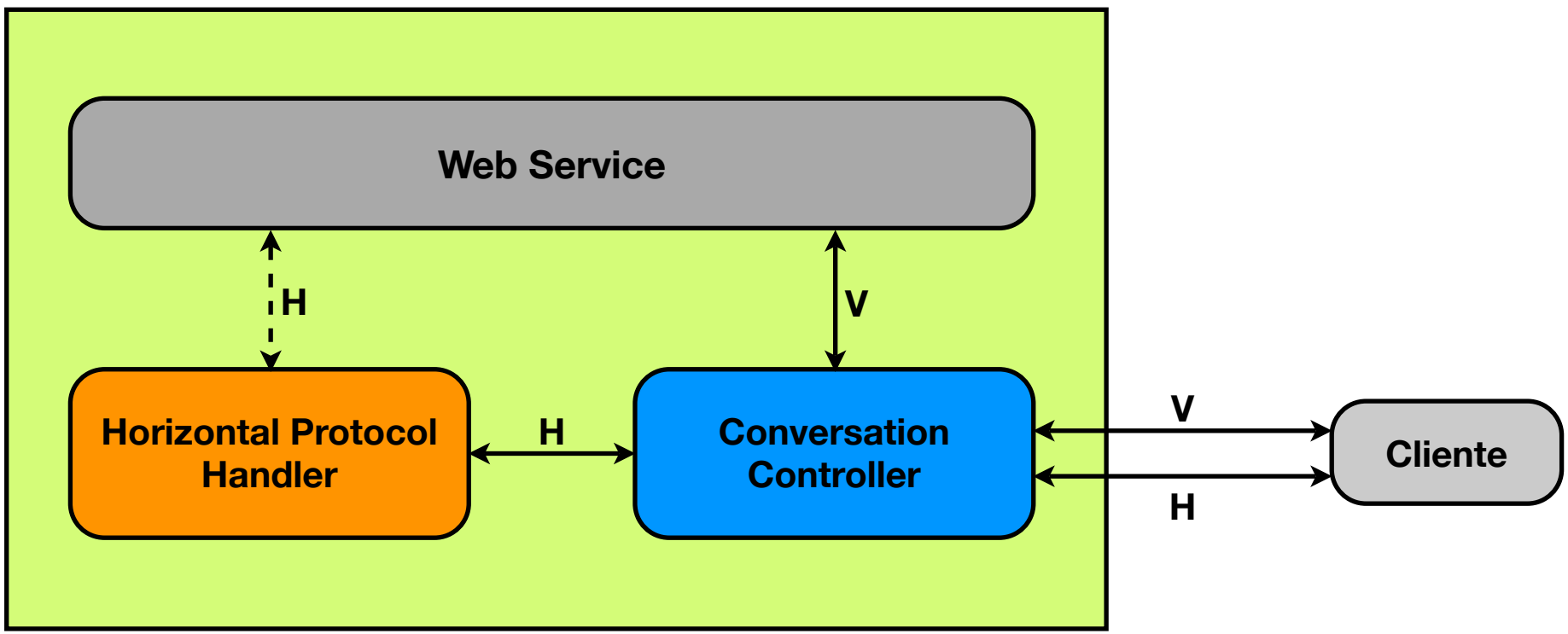
Infra-estrutura para Protocolos de Coordenação (3)

- Um *protocol handler* pode suportar a execução de protocolos de duas formas:
  1) Recebe, interpreta e envia mensagens do protocolo automaticamente, sem intervenção do Web Service:
     - A implementação do protocolo é transparente para o Web Service
     - Exemplo: entrega fiável de mensagens; toda a lógica de armazenamento, *timeouts*, retransmissões, confirmações, etc. é interna à infra-estrutura
  2) Partilha a operação do protocolo com o Web Service:
     - A infra-estrutura oferece primitivas de suporte ao protocolo (e.g. *prepare, commit, abort*, para garantias transaccionais)
     - O Web Service é responsável pelas decisões (i.e. invocar aquelas primitivas) pois é ele que tem o conhecimento necessário sobre o que se está a passar

Coordenação de Web Services

Infra-estrutura para Protocolos de Coordenação (4)

- Esquema de um *protocol handler:*
  - A interacção Web Service <-> *Protocol Handler*, a tracejado, pode, ou não, existir



V: Conversa de um protocolo de negócio (vertical)
H: Conversa de um protocolo horizontal

Coordenação de Web Services

Infra-estrutura para Protocolos de Coordenação (5)

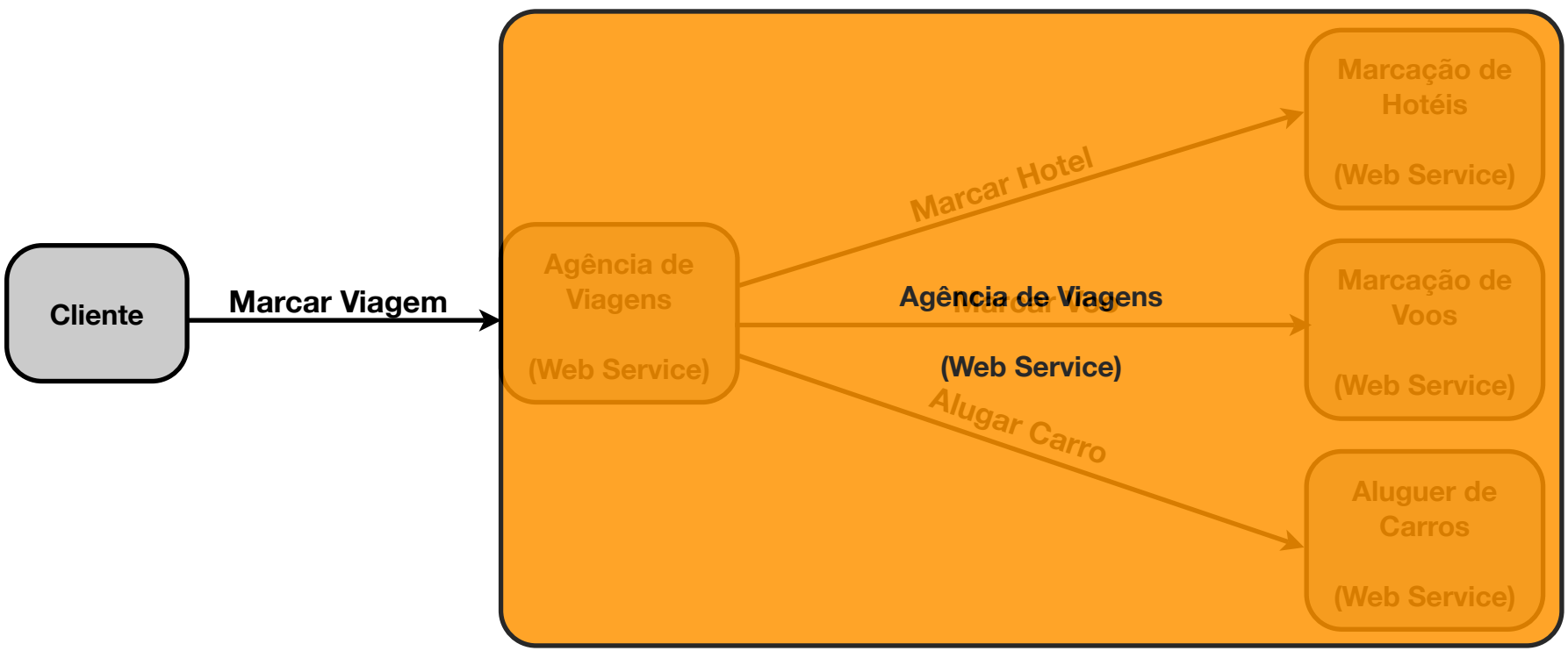
- A infra-estrutura de Web Services pretende interligar um conjunto de serviços prestados por diversas entidades (e.g. vários departamentos, várias empresas de diferentes áreas)
- Para o conceito da coordenação funcionar, há a necessidade de criar *standards* que efectivamente sejam adoptados por todos os intervenientes a vários níveis:
  - Geração e transporte de identificadores únicos das conversas nas mensagens (e.g. no cabeçalho SOAP), para mapear mensagens a conversas
  - Protocolos de controlo (meta-protocolos) que definem os protocolos do sistema a ser executados e como são coordenados
  - Protocolos horizontais padrão para que as garantias dadas por estes se mantenham ao longo de todo o sistema
- Sem *standards*, estas tarefa de implementação estão a cargo de **cada** Web Service no sistema

Composição de Web Services

Composição de Web Services

Motivação

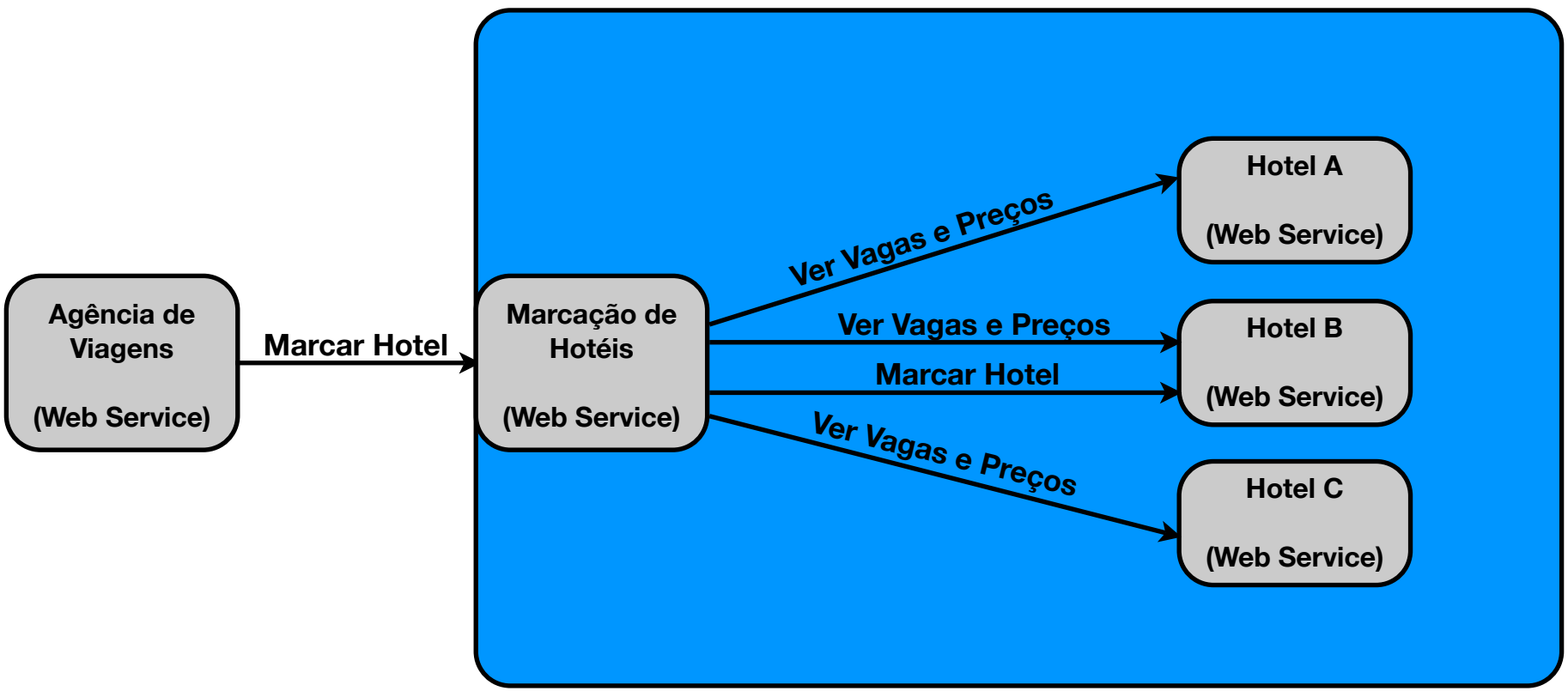
- O objectivo conseguido por um interveniente num conjunto de operações coordenadas entre Web Services, pode ser vista como uma operação complexa, ou composta:
  - Por exemplo, uma agência de viagens pode querer marcar, para cada cliente que vai de férias, um voo, um hotel e um carro de aluguer:



Composição de Web Services

Motivação (2)

- A composição pode ser recursiva:
  - Um Web Service composto pode ser usado por outros Web Services maiores como componente:



Composição de Web Services

Motivação (3)

- Ao contrário da composição tradicional, a composição de Web Services não implica a integração física ou administrativa dos componentes.
  - A fronteira de um Web Service composto é apenas lógica, já que os seus componentes podem ter localizações geográficas dispersas e estar sob a alçada de diferentes domínios administrativos (e.g. diferentes departamentos, diferentes empresas)
- As linguagens de programação convencionais não foram desenhadas tendo a composição em mente.
  - O seu uso para esse fim desvia a maior parte do tempo gasto no desenvolvimento para pormenores de baixo nível, tais como:
    - Conversão de dados
    - Criação de mensagens
    - Rarantir a manutenção de estado persistente durante a execução
    - Ligação dinâmica (*dynamic binding*) de componentes, etc

Composição de Web Services

O que é a Composição?

- A infra-estrutura de composição pretende ser um *Middleware* que resolve aqueles problemas, oferecendo uma *framework* que facilite a definição, criação e execução de Web Services complexos **focando a sua preocupação na lógica de negócio**, e não nos pormenores de baixo nível

- A composição de Web Services foca-se na implementação interna das operações dos serviços
  - A especificação de um serviço composto é interno a uma empresa e é mantido privado
    - Apesar de poder usar outros Web Services externos como seus componentes (exemplo da agência de viagens)

- **É transparente para os clientes**

Composição de Web Services

Relação com a Coordenação

- Os protocolos de coordenação são documentos públicos, definidos por consórcios de *standardização* e através de linguagens padrão, que têm como objectivo suportar a descoberta de serviços em tempo de programação e a ligação a esses mesmos serviços em tempo de execução
  - Os protocolos de composição são especificações privadas que têm como objectivo levar a cabo operações complexas de lógica de negócio que envolvem mais que um Web Service escondendo essa complexidade por detrás de operações simples no Web Service composto
- Os intervenientes dos protocolos de coordenação estão cientes de que estão a executar um protocolo com outros Web Services participantes
  - O Web Service composto esconde a sua complexidade. Um cliente apenas vê a sua interface e não está ciente de que aquele invoca operações noutros serviços como parte da sua lógica interna

Composição de Web Services

Relação com a Coordenação (2)

- Um Web Service composto comunica com outros usando protocolos de coordenação, portanto:
  - A coordenação impõe restrições à forma como a composição se processa
    - A ordem pela qual as operações são invocadas tem que estar de acordo com o protocolo de coordenação
  - A lógica de composição determina as conversas que o serviço composto vai executar

- Num Web Service há, portanto, uma clara relação entre:
  - A sua composição interna
  - A sua coordenação externa

Composição de Web Services

Limitações do Middleware de Composição Tradicional

- O sucesso da composição depende da existência de componentes e modelos de interacção adequados:
  - As suas funções e interfaces devem ser descritas numa linguagem inteligível por todos os que os pretendam usar (i.e. devem seguir *standards*)
- Os sistemas tradicionais de *Middleware* (e.g. WfMS) pecam por pretenderem ser o mais genéricos e flexíveis possível e integrar aplicações completamente heterogéneas
  - Isto requere desenvolvimento *ad-hoc* para cada componente, pelo que grande parte do esforço centra-se nesta parte e não na lógica de negócio em si
- Não existe também um *standard* para um modelo de composição em si

- Os WebServices são uma plataforma mais adequada à composição devido às suas **interfaces bem definidas** e ao **uso de *standards* para os descrever e especificar as suas interacções**

Composição de Web Services

Elementos do Middleware de Composição

- **Modelo de Composição e Linguagem**
  - Permite a especificação da composição:
    - Os serviços que podem ser combinados, a ordem pela qual são invocados, como são construídas as mensagens, etc.
  - À especificação feita através de uma linguagem de composição dá-se o nome de *composition schema*:
    - Este define a lógica de negócio do Web Service composto
    - Pode ser vista como um programa escrito numa linguagem criada para o efeito (por oposição ao uso de linguagens convencionais)

Composição de Web Services

Elementos do Middleware de Composição (2)

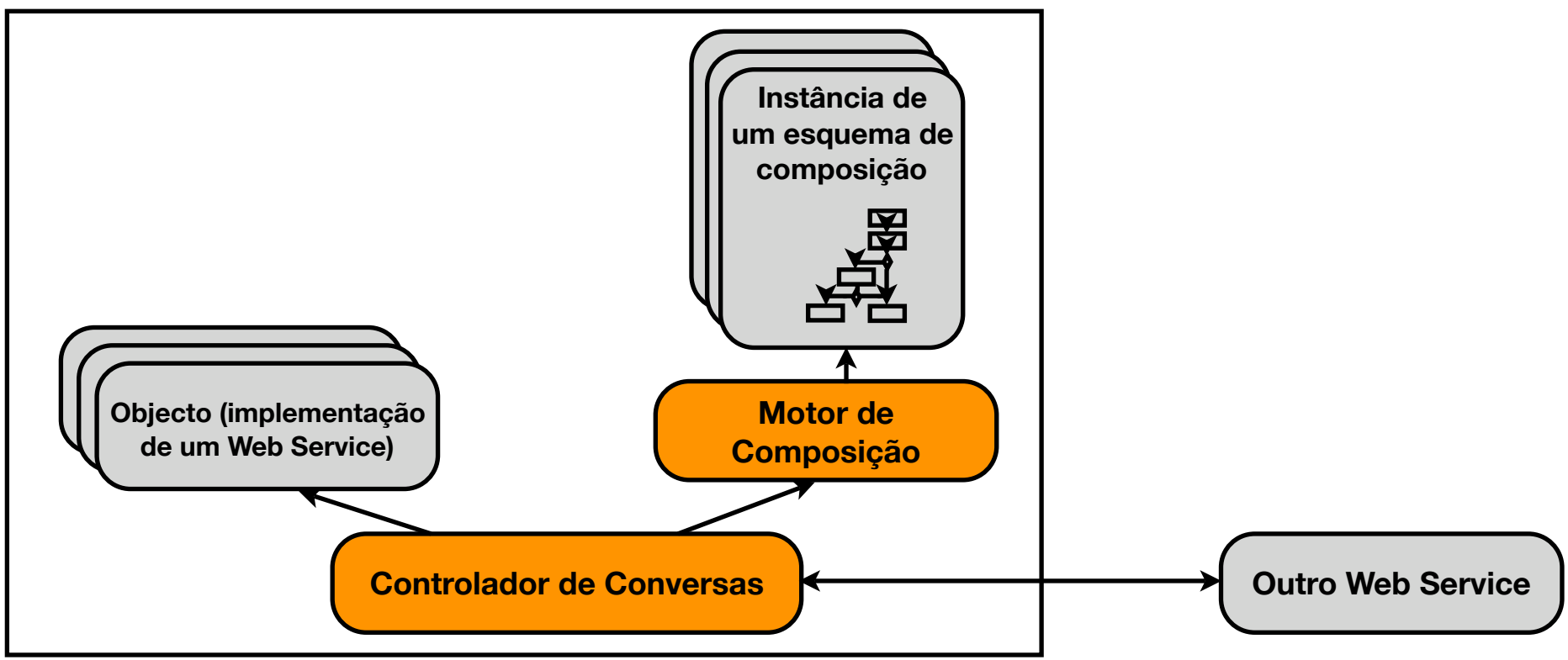
- **Ambiente de Desenvolvimento**
  - Tipicamente tem uma interface gráfica onde os "programadores" podem especificar a composição (o *composition schema*) arrastando e largando representações gráficas dos Web Services a compor e criando grafos para definir a ordem de invocação.
  - Esta informação gráfica é traduzida, pela ferramenta de desenvolvimento, para uma descrição textual da especificação (e.g. documento XML na linguagem BPEL4WS)

- **Ambiente de Execução (ou *Composition Engine*)**
  - É o motor que executa a lógica de negócio especificada no Ambiente de Desenvolvimento e descrita na Linguagem de Composição, invocando as operações definidas por aquela.
  - Cada execução de um serviço composto é denominada uma instância de composição (*composition instance*). Podem existir várias instâncias de serviços compostos a executar no mesmo motor simultaneamente

Composição de Web Services

Motor de Composição e Controlador de Conversas

- O facto de poderem existir várias instâncias de Web Services compostos a executar no mesmo ambiente de composição introduz um problema de encaminhamento análogo ao já encontrado na coordenação e resolvido pelos Controladores de Conversa.
- Um sistema pode incluir um controlador de conversa e um ambiente de composição conjuntamente:



Composição de Web Services

Dimensões de um Modelo de Composição

- **Modelo de Componentes**
  - Define a natureza dos elementos a serem compostos em termos das suas características
- **Modelo de Orquestração**
  - Define abstracções e linguagens usadas para a definição da ordem pela qual os serviços são invocados
- **Modelo de Dados e Transferência a Dados**
  - Define como os dados são especificados e como são trocados entre componentes
- **Modelo de Selecção de Serviços**
  - Define como a ligação entre serviços (estática ou dinâmica) é feita, i.e. como um serviço específico é seleccionado como componente
- **Modelo de Transacções**
  - Define qual a semântica transaccional associada à composição
- **Modelo de Tratamento de Excepções**
  - Define como devem ser tratadas as situações excepcionais na execução do serviço composto para que esta não seja abortada

Composição de Web Services

Exemplo: BPEL4WS

- Business Process Execution Language for Web Services (BPEL4S)
- Proposta de uma linguagem *standard* para especificar protocolos de coordenação e composição de Web Services
- Criada por um consórcio composto por BEA, Microsoft e IBM
- Primeira versão em 2002, revista em Março de 2003
- Assume que os serviços a compor e coordenar estão descritos usando WSDL

- As especificações XML BPEL4WS para um serviço composto são documentos que definem:
  - Os papeis dos participantes na troca de mensagens
  - Os tipos de portos que devem ser suportados pelo próprio serviço
  - A orquestração e os outros aspectos que fazem parte da definição do serviço
  - Informação de correlação que define como as mensagens devem ser encaminhadas para os componentes correctos

Composição de Web Services

Modelo de Componentes

- Define o tipo de componentes a suportar pela composição e os pressupostos sobre as características que devem apresentar:
  - Num extremo, o modelo pode assumir apenas componentes que implementem um subconjunto dos *standards* dos Web Services:
    - e.g. HTTP + SOAP + WSDL + WS-Transaction
    - Isto limita a heterogeneidade logo torna a composição mais fácil
  - No outro extremo podem ser feitos apenas pressupostos muito simples sobre os componentes:
    - e.g. só assumir que os componentes interagem através da troca de mensagens XML de forma síncrona (como RPC's)
    - A vantagem é que torna o sistema extremamente genérico e flexível
    - A desvantagem é que torna o trabalho de composição muito mais complicado devido à heterogeneidade dos componentes

Composição de Web Services

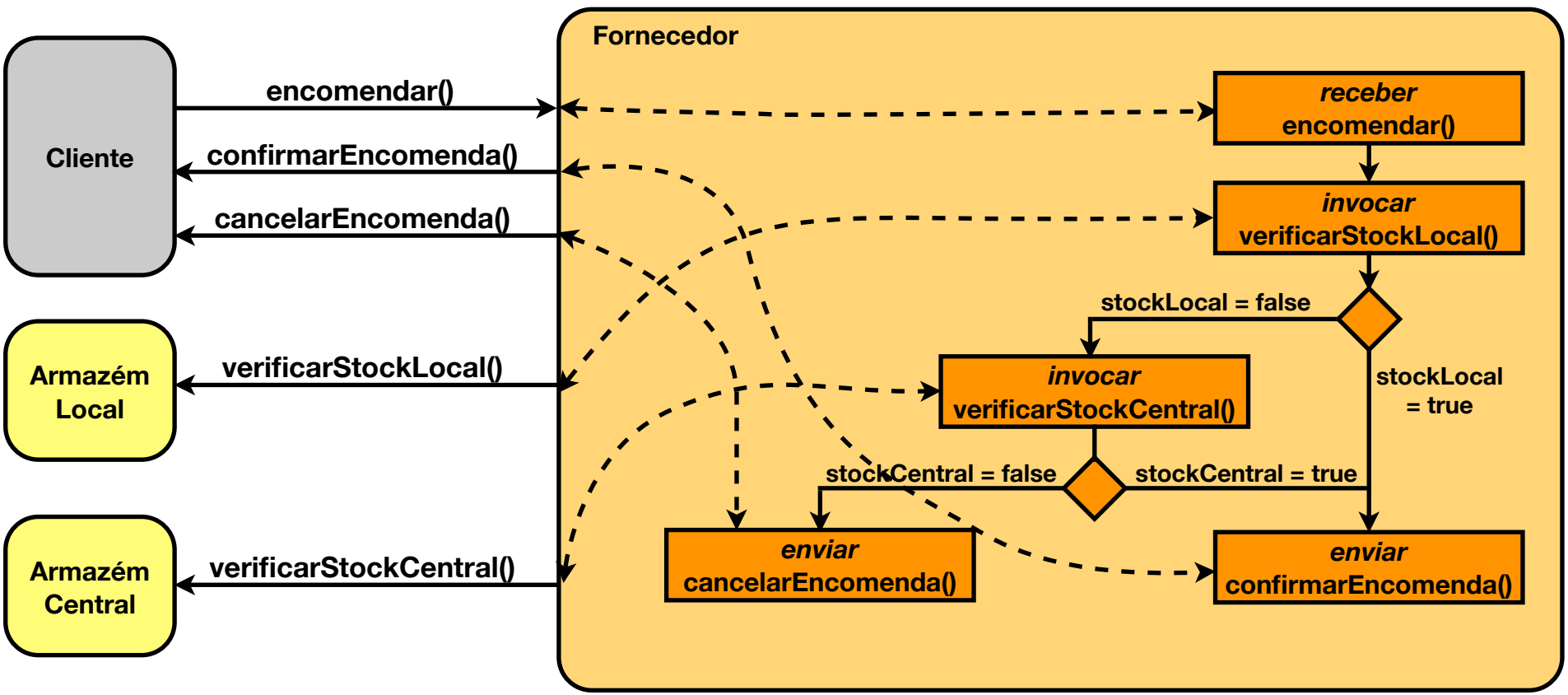
Modelo de Componentes do BPEL4WS

- Baseado em actividades
  - Actividades básicas representam os componentes em si e correspondem à invocação de uma operação WSDL
    - Estão disponíveis operações de invocação (*invoke*), recepção (*receive*) e resposta (*reply*)
  - Actividades estruturadas são usadas para definir a orquestração
  - Outras actividades:
    - Atribuição de dados a variáveis (*assign*)
    - Esperas temporizadas (*wait*)

Composição de Web Services

Modelo de Orquestração

- Define como os diferentes serviços são compostos num conjunto coerente. Especifica:
  - A ordem pela qual as operações aos serviços são invocadas
  - As condições pelas quais são tomadas as decisões de invocar ou não uma determinada operação



Composição de Web Services

Modelo de Orquestração (2)

- As operações indicadas neste modelo coincidem com operações que se podem executar sobre Web Services, nomeadamente:
  - Invocações síncronas (pedido/resposta), representadas no diagrama por *invocar*
  - Notificações a outros Web Services, representadas no diagrama por *enviar*
  - Recepção de mensagens correspondentes à invocação de uma operação da interface por parte de outro serviço, representadas no diagrama por *receber*
  - Se uma mensagem recebida faz parte de uma interacção tipo pedido/resposta, irá existir a respectiva resposta representada por *responder*

- São também incluídos, tipicamente, outros pormenores como a definição dos serviços para onde cada mensagem é enviada, a especificação de como construir as mensagens com base nas actividades anteriores, etc.
  - Esta descrição inclui as condições e a descrição dos dados, ao contrário das especificações de protocolos de coordenação

Composição de Web Services

Modelo de Orquestração do BPEL4WS

- Baseado nas actividades estruturadas:
  - Agrupam um conjunto de outras actividades básicas ou estruturadas, estabelecendo uma ordenação entre elas.
  - Existem os seguintes tipos de actividades estruturadas:
    - *Sequence*: conjunto de actividades a executar sequencialmente
    - *Switch*: escolha entre actividades baseada nas suas condições (como em C)
    - *Pick*: Inclui um conjunto de eventos (e.g. recepção de uma mensagem, *timeout*), cada um associado a uma actividade. Quando um evento é recebido, a respectiva actividade é executada e o *pick* é considerado completo
    - *While*: Inclui uma única actividade (básica ou estruturada) que é executada repetidamente enquanto uma determinada condição é verdadeira
    - *Flow*: Agrupa um conjunto de actividades a executar em paralelo. É considerado completo quando todas as actividades terminam

Composição de Web Services

Modelo de Dados e Transferência de Dados

- Tal como nas linguagens de programação, os modelos de composição de serviços necessitam de formas explícitas para definir dados e o acesso a estes

- Os tipos de dados podem ser divididos em:
  - Dados específicos da aplicação: os parâmetros recebidos ou enviados como parte da troca de mensagens (e.g. na invocação de *verificarStockLocal()*, serão passados parâmetros relativos ao tipo de produto, quantidade, etc.)
  - Dados de controlo: os parâmetros relevantes para avaliar condições (e.g. na resposta à invocação *verificarStockLocal()*, o parâmetro *stockLocal* vai ditar qual a actividade seguinte no fluxo
- Na maior parte dos casos, os dados da aplicação têm conteúdo mais rico que os de controlo:
  - A informação de controlo necessária é apenas um subconjunto de todos os dados e tipicamente restrita a tipos simples (e.g. string, inteiro, real)
  - A informação de controlo pode ser tipicamente derivada dos dados da aplicação, de acordo com um mapeamento pré-definido

Composição de Web Services

Modelo de Dados e Transferência de Dados (2)

- Há dois métodos de tratar os dados da aplicação:

  1) Tratar os dados com uma *caixa negra*, passando apenas ponteiros para estes entre as actividades
     - Em vez de trocarem documentos, os Web Services só trocam URL's
     - Muitos processos envolver troca de documentos complexos, quase nunca modificados quando passados entre actividades
     - Do ponto de vista da composição, o URL são os dados aplicacionais, logo o modelo pode ignorar trocas complexas de dados entre actividades
     - O ambiente de composição não tem que lidar com enormes quantidades de dados
     - Muitas aplicações esperam receber os dados em si e não um ponteiro, pelo que os programadores têm que escrever adaptadores para resolver este problema

Composição de Web Services

Modelo de Dados e Transferência de Dados (3)

2) Tornar todos os dados da aplicação explícitos

- A descrição da composição inclui a especificação de todos os dados trocados, de forma semelhante ao que é feito nos dados de controlo
- O ambiente de composição tem que lidar com grandes e complexas quantidades de dados.
- Se os dados forem demasiado grandes ou complexos, o sistema pode não ter a capacidade de os tratar e pode deixar de funcionar

- Em termos de tipos de dados, estes podem ser todos representados em XML ou incluir alguns documentos binários
  - O formato XML exige um conjunto de passos de interpretação (parsing) adicionais que, para certos documentos, se pode tornar demasiado ineficiente.
  - Podem ser incluídos alguns documentos binários, num qualquer formato proprietário, para melhorar o desempenho do sistema. Estes podem ser, por exemplo, enviados como anexos binários a mensagens SOAP.

Composição de Web Services

Modelo de Dados e Transferência de Dados (4)

- Para a transferência de dados, existem também dois métodos fundamentais:

  1) *Blackboard*

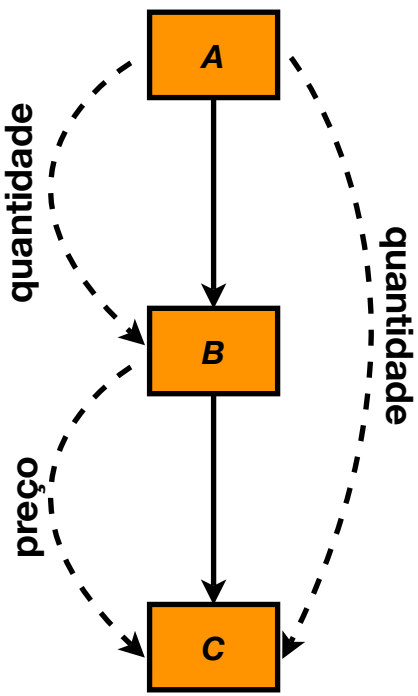
  - É baseado no princípio em que todos os dados são explicitamente nomeados e listados (como nas linguagens de programação tradicionais)
  - O *blackboard* é uma colecção de variáveis onde as actividades (invocações a operações) depositam conteúdo
  - Uma mensagem a ser enviada usa como parâmetros valores de variáveis do *blackboard*, assim como os parâmetros de uma mensagem recebida são colocados nas respectivas variáveis
  - As modificações podem (e devem) ser feitas de forma atómica
  - Tal como um processo de um S.O., cada instância do ambiente de composição tem o seu *blackboard*
  - Este é o método mais natural para os programadores devido às semelhanças com as linguagens de programação tradicionais

Composição de Web Services

Modelo de Dados e Transferência de Dados (5)

2) Fluxo explícito de dados

- Os dados trocados entre actividades são definidos explicitamente durante o desenvolvimento da composição
- O desenhador pode definir, por exemplo, que os dados de entrada de uma determinada actividade (e.g. *C*) são os de saída de outra (e.g. *quantidade* na actividade *A*), não necessariamente a que foi executada imediatamente antes
- Esta aproximação é mais rica e flexível mas introduz complexidade adicional na especificação da composição
- Cria dependências de controlo implícitas no sistema, dado que as actividades que são fonte de informação para outras têm que terminar primeiro



Composição de Web Services

Modelo de Dados e Transferência de Dados do BPEL4WS

- O BPEL4S mantém o estado de cada processo e manipula os dados de controlo usando variáveis:
  - São análogas às variáveis das linguagens de programação convencionais:
    - Caraceterizam-se por nome e tipo de dados
    - O tipo de dados é uma referência para um tipo de mensagem WSDL, um tipo ou um elemento *XML Schema*
  - Quando definidas, podem ser usadas como parâmetros de entrada ou saída em invocações de operações
  - Aquando da criação da instância do serviço composto, as variáveis não são inicializadas
    - Se forem actualizadas (e.g. usando-as como parâmetro de output ou através de uma expressão usando a operação *assign*) são inicializadas implicitamente
  - A transferência de dados, neste modelo, segue o modelo *blackboard*

Composição de Web Services

Modelo de Selecção de Serviços

- Para executar a lógica de composição, o motor tem que saber que serviço específico (e.g. o seu URL) é alvo de cada mensagem
  - Esta informação é tipicamente especificada de forma abstracta no modelo de composição: as linguagens normalmente compõem tipos de portos e não os portos em si
- Há quatro formas principais de seleccionar os serviços a utilizar em cada caso:
  - **Ligação estática** (*static binding*)
    - É a maneira mais simples: os URI's estão embutidos na especificação
  - **Ligação dinâmica por referência** (*dynamic binding by reference*)
    - As actividades determinam os URI's dos serviços usando valores de variáveis pré-definidos
    - Não são feitos pressupostos sobre como essas variáveis são preenchidas, pode ser, por exemplo, por uma qualquer outra operação anterior

Composição de Web Services

Modelo de Selecção de Serviços (2)

- **Ligação dinâmica por procura** (*dynamic binding by lookup*)
  - O Middleware de composição permite a definição, para cada actividade, de uma *query* cujo resultado é usado para determinar o serviço a invocar
  - Por exemplo, a linguagem WSFL (um antecessor do BPEL4WS) permite incluir uma *query* desse tipo a um repositório UDDI dentro da especificação de cada actividade (usando a API que o UDDI fornece)
- **Selecção dinâmica de operações** (*dynamic operation selection*)
  - Permite que, não só o serviço, mas também a própria operação a invocar no serviço, sejam determinados dinamicamente
  - Modelo já existente no CORBA
  - Por exemplo, uma agência de viagens pode invocar operações diferentes em serviços diferentes consoante o cliente escolha ir de barco, avião ou comboio
    - A escolha pode ser definida ao nível da orquestração ou, para reduzir a complexidade, o Middleware pode oferecer a capacidade de criação de actividades abstractas, que escolhem as operações em tempo de execução

Composição de Web Services

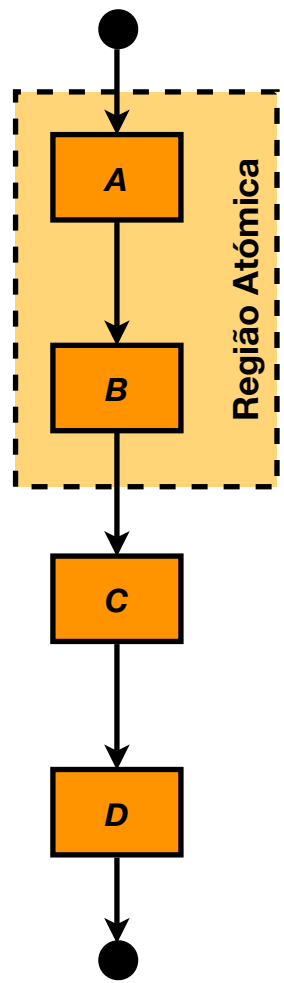
Modelo de Selecção de Serviços do BPEL4WS

- A selecção de serviços no BPEL4WS introduz três noções fundamentais:
  - ***Partner Link Type***: relaciona um par de papeis que trocam mensagens durante a execução do processo e os tipos de portos que os serviços que tomam esses papeis devem implementar
  - ***Partner Link***: define que serviços tomam os papeis de um *partner link type*, ou seja, qual o papel a tomar pelo processo e qual o papel a tomar pelo outro Web Service que irá interagir. Não especifica qual o outro serviço concretamente
  - ***Endpoint Reference***: Associa um serviço concreto a um *partner link*. Essa associação pode ser feita em tempo de configuração (*deployment*) ou em tempo de execução
- As definições das actividades podem então referenciar *partner links* para que o motor de composição saiba para que serviço deve enviar cada mensagem

Composição de Web Services

Modelo de Transacções

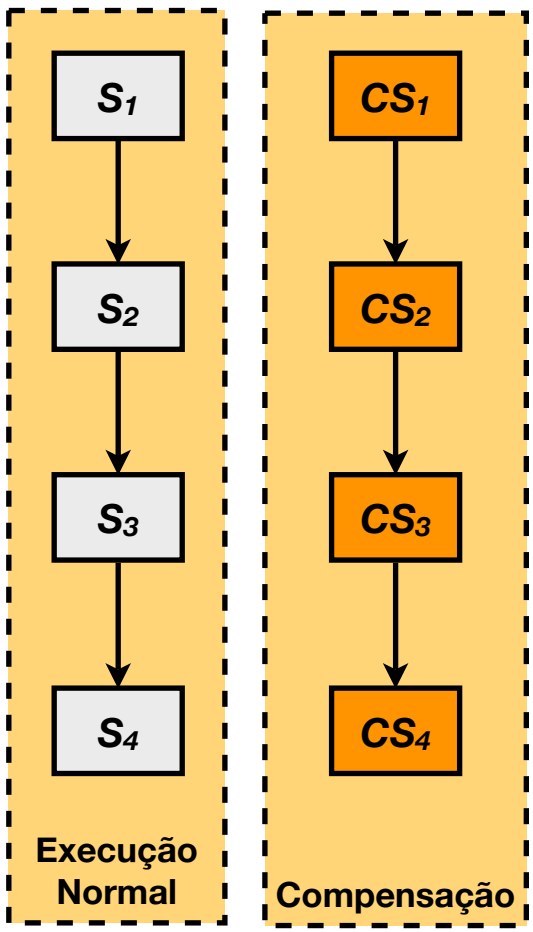
- A forma que o *Middleware* tem de oferecer comportamento transaccional aos serviços compostos é permitir a definição de regiões atómicas dentro do modelo de orquestração
- A atomicidade pode ser conseguida usando protocolos *2-phase commit*, possivelmente baseados no protocolo de coordenação horizontal WS-Transaction
- Este comportamento pode ser totalmente implementado pelo *Middleware* para que o "programador" do serviço composto não se tenha que preocupar com este problema
- Contudo, a composição de Web Services, devido à sua natureza distribuída aos níveis geográfico e administrativo, necessita de uma semântica transaccional mais fraca (e.g. não é possível ter recursos adquiridos - *locked* - durante longos períodos)
  - Nestes casos, é adoptada uma solução de compensação, em que é possível desfazer (undo) operações executando outras
  - Do ponto de vista da composição isto significa que, quando existe uma falta, o Middleware toma medidas correctivas sob a forma de operações pré-definidas para o efeito



Composição de Web Services

Modelo de Transacções (2)

- Nem todos os serviços suportam WS-Transaction ou o "programador" do Web Service composto pode querer definir explicitamente a lógica de negócio relativa à compensação
  - Para suportar este tipo de situações, a maioria dos modelos de composição (e.g. o BPEL4WS) oferece abstracções que permitem que as transacções de longa duração sejam divididas em sub-transacções
  - Cada sub-transacção goza também das propriedades ACID
  - Para cada sub-transacção S_k é especificada uma sub-transacção de compensação CS_k que a desfaz
  - Muitos sistemas permitem que esta lógica de compensação seja definida na forma de um esquema de orquestração que descreve como cada região atómica deve ser compensada



Composição de Web Services

Modelo de Transacções do BPEL4WS

- É usado o modelo das transacções de compensação:
  - É possível definir a lógica necessária para desfazer um determinado conjunto de operações
  - A lógica de compensação é especificada por um ***compensation handler*** que consiste numa única actividade (básica ou estruturada)
  - Os *compensation handlers* são incluídos na lógica de tratamento de excepções do BPEL4WS

Composição de Web Services

Modelo de Tratamento de Excepções

- Uma excepção é um desvio do fluxo execução esperado num determinado momento
  - São tipicamente causadas por falhas no sistema ou em aplicações invocadas (e.g. perde-se a ligação com o servidor)
  - Podem também ser situações previstas na semântica do Web Service mas que são infrequentes (e.g. um cliente cancela uma encomenda feita previamente)
- Uma forma de lidar com excepções é através de transacções
  - Não é a melhor forma pois obriga sempre a que parte da execução anterior seja perdida e nem sempre é isso que se pretende
- Outras formas mais refinadas de tratar estas situações excepcionais são:
  - Tratamento baseado no fluxo
  - Tratamento baseado em *try-catch-throw*
  - Tratamento baseado em regras

Composição de Web Services

Modelo de Tratamento de Excepções (2)

- **Tratamento de excepções baseado no fluxo**
  - Quando não existem primitivas especializadas para o efeito, os programadores podem usar a mesma técnica adoptada em algumas linguagens de programação de terceira geração (e.g. C):
    - No final de cada operação, o seu resultado é analisado à procura de erros (e.g. verificar o valor de retorno); caso algum seja detectado é tomada a acção apropriada
  - Isto pode ser concretizado na composição de serviços através da introdução de condições no modelo de orquestração para o efeito
  - As excepções que se referem a operações que não retornam (por oposição aos resultados incorrectos) têm que ser tratadas introduzindo *timeouts* nas actividades. Quando estes expiram, a actividade prossegue tomando alguma acção correctiva se necessário

Composição de Web Services

Modelo de Tratamento de Excepções (3)

- **Tratamento de excepções baseado em *try-catch-throw***
  - Conceptualmente semelhante às instruções Java com o mesmo nome
  - Inclui a lógica de tratamento de excepções na execução da própria actividade:
    - A excepção está associada a uma determinada condição incluída nos dados da composição do serviço. Quando esta se verifica, a execução passa à parte relativa ao tratamento da excepção; caso contrário segue o seu fluxo normal
  - Permite uma separação clara entre a lógica de execução “normal” e a das excepções

Composição de Web Services

Modelo de Tratamento de Excepções (4)

- **Tratamento de excepções baseado em *regras***
  - A lógica de tratamento de excepções é especificada em termos de regras E-C-A (*on Event, if Condition then Action*)
    - O evento define um evento excepcional a ser recebido na forma de mensagens (e.g. cliente cancela encomenda) ou timeouts
    - A condição é uma expressão booleana sobre dados da mensagem que determina se efectivamente esta corresponde a uma situação de excepção
    - A acção é a reacção à excepção (e.g. invocar operações ou abortar transacções)
  - As regras são tipicamente especificadas numa linguagem textual que enriquece a representação gráfica do modelo de orquestração
  - Permite uma separação clara entre os comportamentos normal e excepcional do sistema à custa da interpretação de mais linguagens
  - O conjunto final de regras de um sistema é complexo e difícil de analisar

Composição de Web Services

Modelo de Tratamento de Excepções do BPEL4WS

- O BPEL4WS segue o modelo *try-catch-throw*
  - Cada actividade (básica ou estruturada) define um foco de execução
  - Cada elemento que define um foco de execução pode incluir a especificação de um ou mais elementos de tratamento de faltas (*fault handlers*)
  - Um *fault handler* é caracterizado por um elemento *catch* que define a falta que trata e a actividade (básica ou estruturada) que deve executar em caso de ocorrência daquela
  - As faltas podem ser geradas durante a execução de uma actividade dentro do seu foco, quer pela operação invocada (pelo retorno de uma mensagem de falta WSDL) ou pelo próprio motor de execução (devido a qualquer erro de execução)
    - As faltas podem ser também explicitamente geradas no esquema de orquestração usando uma actividade *throw*

Composição de Web Services

Modelo de Tratamento de Excepções do BPEL4WS (2)

- Quando uma falta ocorre num determinado foco de execução, o motor BPEL4WS pára todas as actividades que estão a executar e arranca a actividade definida no respectivo *fault handler*
  - Todos os focos têm um *fault handler* por omissão. Se nenhum outro existir este é executado
- Outra forma de tratar excepções são os *event handlers*. Estes permitem uma monitorização contínua de um determinado evento (e.g. *timeout* ou recepção de uma mensagem), e executam uma actividade pré-definida aquando da sua ocorrência

- Relativamente ao comportamento transaccional, os *compensation handlers* são associados a cada foco de execução.
  - Cada foco tem, também, um *compensation handler* por omissão que consiste apenas em invocar os *compensation handlers* de cada foco contido no actual

Composição de Web Services

Encaminhamento para as Instâncias no BPEL4WS

- Um problema importante da composição de serviços é o encaminhamento das mensagens recebidas para as instâncias de serviços compostos (a executar no ambiente de composição) correctas
  - O BPEL4WS define, no seu esquema de composição, o conceito de ***correlation sets***:
    - Identificam um conjunto de dados
    - Podem ser associados a mensagens enviadas ou recebidas dentro das actividades *invoke*, *reply* ou *receive*
    - Associando um par de mensagens ao mesmo *correlation set*, o programador define que, se têm os mesmos valores para o *correlation set*, então pertencem à mesma instância

Referências

- G. Alonso et al., *Web Services - Concepts, Architectures and Applications*, Springer Verlag, 2004

- D. F. Ferguson et al., *Secure, Reliable, Transacted Web Services: Architecture and Composition*, IBM & Microsoft, Setembro 2003

- S. Barkodar, A. Stashkova, *A Simple Synchronous BPEL Process*, Sun NetBeans 5.5 Knowledge Base, Outubro 2006

- B. May, D. Markovsky, *Understanding the Travel Reservation Service*, Sun NetBeans 5.5 Knowledge Base, Outubro 2006