Internet das Coisas nos processos de negócio

Carlos Cândido*, Dulce Domingos, and Francisco Martins

LaSIGE, Faculdade Ciências da Universidade de Lisboa,
Edifício C6, Piso 3, Campo Grande
1749-016 Lisboa, Portugal
{ccandido}@lasige.di.fc.ul.pt
{dulce,fmartins}@di.fc.ul.pt
http://lasige.di.fc.ul.pt/

Resumo Os processos de negócio podem beneficiar de forma significativa do desenvolvimento da Internet das Coisas, já que estes podem utilizar a informação de contexto disponibilizada pelos sensores para optimizar a sua execução e para reagir a situações emergentes em tempo real. Actualmente, a WS-BPEL é a norma de-facto utilizada para definir processos através da composição de serviços web. No entanto, esta linguagem, sendo baseada num paradigma de orquestração de serviços, limita a forma como os processos podem usufruir da informação de contexto. Neste artigo é apresentada uma extensão para a WS-BPEL de modo a incluir variáveis de contexto. Estas variáveis mantêm os valores dos sensores. A sua atualização é efectuada através do paradigma publicação/subscrição. A extensão é concretizada através de um mecanismo de transformação da linguagem, tornando-a independente do motor de execução. Este trabalho constitui a base para suportar processos de negócio mais reactivos, o qual permitirá estender a linguagem com novos construtores, como por exemplo, o *when-then*.

Keywords: Internet das Coisas, Processos de negócio, WS-BPEL, publicação/subscrição

1 Introdução

A Internet das Coisas (IoT, do inglês Internet of Things) pretende aproximar o mundo real aos sistemas de informação. Em particular, nos processos de negócio, a IoT apresenta-se como uma vantagem em termos competitivos, já que a informação de contexto disponibilizada pelos sensores pode ser utilizada para optimizar a sua execução e para permitir a sua adaptação em tempo real a alterações do ambiente.

Actualmente, a WS-BPEL [1] é a norma de-facto utilizada para definir processos através da composição de serviços web. Tendo em vista a integração da IoT nos processos de negócio, assistimos à disponibilização da sua informação

* Este trabalho foi financiado pela FCT através do projecto PATI (PTDC/EIA-EIA/103751/2008) e através do programa de financiamento multianual do LaSIGE.

através de serviços web, os quais podem ser suportados directamente nos sensores ou através de middleware [2] . Esta abordagem apresenta a vantagem adicional de encapsular as especificidades dos vários tipos de sensores. Sendo a WS-BPEL baseada na orquestração de serviços, a informação dos sensores disponibilizada através de serviços web pode ser utilizada facilmente nos processos utilizando um paradigma síncrono de pedido/resposta.

No entanto, com este paradigma, se pretendermos que um processo tenha informação actualizada sobre as alterações que ocorrem no ambiente, o processo terá de periodicamente obter a informação dos sensores, sobrecarregando o motor de execução dos processos e aumentando o número de mensagens trocadas.

O trabalho apresentado neste artigo pretende simplificar a utilização de informação de contexto actualizada nos processos de negócio. Com este objectivo foi definida uma extensão à WS-BPEL de modo a incluir variáveis de contexto. Estas variáveis são actualizadas de forma assíncrona através do paradigma de subscritor/publicador. A comunicação entre as instâncias de processos e os sensores é efectuada de acordo com a norma WS-Notifications. A concretização da extensão é efectuada em tempo de modelação, facto que apresenta a vantagem de a tornar independente do motor de execução de processos. Este trabalho constitui a base para suportar processos de negócio mais reactivos, o qual permitirá estender a linguagem com novos construtores, como por exemplo, o *when-then*.

Na secção seguinte é descrito um caso de uso. Na secção 3 é apresentado um breve resumo das normas utilizadas neste trabalho. O trabalho relacionado é apresentado na secção 4 e, na secção 5, é apresentado o trabalho desenvolvido. Finalmente são apresentadas as conclusões e o trabalho futuro.

2 Caso de uso

De seguida é exposto um caso de uso onde se evidencia o uso da internet das coisas num processo de negócio.

Uma empresa de distribuição recebe um contentor de morangos num caís de descarga. Após a descarga este deve ser colocado num camião e transportado para a central de distribuição localizada a 200 quilómetros para então ser feita a distribuição pelas lojas de retalho. O contentor em causa está equipado com sensores e durante o transporte estes vão notificando o estado do contentor. Algures no tempo ocorre uma notificação que indica a subida da temperatura dentro do contentor para um valor próximo do valor no qual os morangos vão entrar em biodegradação. Se os morangos forem submetidos à viagem inicialmente definida, quando chegarem ao consumidor provavelmente não estão num estado comestível. Para evitar este desperdício, quando ocorre a notificação da subida de temperatura, é alterada a rota do contentor e este será encaminhado para uma central de distribuição mais próxima que o destino inicialmente traçado de forma a salvaguardar o estado dos alimentos.

Neste caso de uso percebe-se facilmente como pode ser benéfico o uso de informação de contexto para enriquecer o processo de negócio. Baseando-se nas informações disponibilizadas pelos sensores, um processo consegue adaptar-se e

reagir a situações em tempo real o que implica melhorias no seu desempenho. Atualmente o WS-BPEL não fornece mecanismos que permitam programar de forma simples esta interação. A nossa solução tem como base criar esses mecanismos como apresentado mais à frente.

3 Conceitos

Nesta secção é efetuado um breve resumo da linguagem WS-BPEL. Previamente são apresentadas algumas normas usadas no trabalho apresentado neste artigo.

3.1 Web Services Addressing

Web Services Addressing (WS-Addressing) [3] define mecanismos de transporte-neutro para os serviços Web e a troca de mensagens. Esta especificação define elementos usando Extensible Markup Language (XML) para identificar univocamente pontos terminais (do inglês *Endpoint*) dos serviços Web de forma a garantir o sucesso em troca de mensagens ponto a ponto. Um dos elementos definidos é o *EndPointReference* (EPR) que através do campo *address* representa o endereço de um serviço web. Para além desse campo, um EPR disponibiliza campos opcionais (como o *ReferenceParameters*) para, por exemplo, distinguir EPRs com o mesmo *address*.

3.2 Web Services Description Language

Web Service Description Language (ou apenas WSDL) [4] é uma linguagem baseada em XML para descrever serviços de rede como um conjunto de terminais que operam por troca de mensagens. Um serviço descrito em WSDL inclui o seu nome, o seu endereço (*EndPointReference*), o *Binding* do serviço (define os protocolos usados), as operações disponibilizadas, as mensagens trocadas e as possíveis faltas. Um artefacto WSDL contém as informações necessárias para contactar um serviço web.

3.3 Web Services Notification

Web Services Notification (WSN) [5] define um conjunto de normas que têm como objetivo definir a forma como os serviços Web interagem usando Notificações ou eventos. Definem padrões para comunicação no paradigma de Publicador / Subscritor para serviços Web onde uma entidade pode publicar informações para outras entidades sem as ter de conhecer previamente.

As especificações fornecem interfaces para os intervenientes descritas em WSDL. Destas interfaces destacam-se a do serviço publicador (*NotificationProducer*) e a do serviço que recebe notificações (*NotificationConsumer*). Estas especificam as operações mínimas que um serviço publicador deve disponibilizar (*Subscribe* e *GetCurrentMessage*) e o mesmo para o serviço que recebe as notificações (*Notify*). Numa invocação da operação *Subscribe* um subscritor deve indicar, entre outras coisas, o EPR para onde devem ser enviadas as notificações. Quando ocorre uma subscrição é gerado um EPR que identifica essa subscrição.

3.4 Web Services Business Process Execution Language

Web Services Business Process Execution Language (WS-BPEL) [1] é a norma da OASIS para descrever processos de negócio através de composição de serviços web. Um processo de negócio é composto por dois elementos: um ficheiro escrito em Web Service Description Language (WSDL) que descreve as funcionalidades fornecidas pelo processo de negócio (as estruturas de dados trocadas nas mensagens, os endereços dos serviços, etc.) e um ficheiro escrito em WS-BPEL que descreve o processo de negócio.

Um processo de negócio é descrito através de composição de atividades. Existem atividades de controlo de fluxo (*If*, *While*, *Scope*, *Flow*, etc.), atividades de comunicação (*Receive*, *Reply*, *Invoke*, etc.), atribuição de valores (*Assign*), gestão de faltas (*Throw*, *Rethrow*, etc), entre outras. É possível declarar variáveis de qualquer tipo primitivo, complexo (compostos por vários tipos de dados primitivos) e mensagens. As variáveis de mensagens são usadas quase exclusivamente em atividades de comunicação. As variáveis podem ser globais ou locais se declaradas num *Scope*.

Os *PartnerLinks* servem para as atividades de comunicação saberem que operações representam ou devem invocar. Um *PartnerLink* está associado a um *PartnerLinkType* que por sua vez declara papéis (*Roles*) associados a tipos de portas. Ao ser declarado, um *PartnerLink* indica qual o seu papel e/ou qual o papel do segundo interveniente. Uma atividade de comunicação de *input* seleciona uma operação da porta associada ao *MyRole* e uma atividade de *output* seleciona uma operação do *PartnerRole*.

De modo a distinguir as instâncias usa-se o mecanismo *Correlation*. Um *correlationSet* é definido por 1) o tipo de dados simples que vai ser usado e 2) conjunto de regras (uma por tipo de mensagem). Após estar definido, este é associado às atividades de comunicação onde pretendemos utilizar com o *correlationSet*. Cada *correlationSet* só deve ser inicializado uma vez e, em caso de ser usado num *Invoke*, deve-se definir se o *Correlation* é feito no envio, na resposta ou em ambos os sentidos. As propriedades dos *CorrelationSets* definem, através de XPATH, os elementos das mensagens trocadas pelo processo que identificam cada conversação (ie, cada instância).

A norma WS-BPEL prevê a possibilidade de ser extendida. Com este objetivo foram definidos os seguintes mecanismos:

- Atividades de extensão: São atividades novas que podem ter o comportamento que o programador desejar. São declaradas no processo dentro do Construtor "ExtensionActivity" fornecido pela linguagem.
- Operação de Assign estendida: Adicionam comportamento a uma atividade de atribuição.
- Atributos de extensão: Adicionam novos conceitos através de novos atributos nos construtores.
- Construtores/Elementos de extensão: Adiciona novos conceitos através de novos construtores.

Todas as extensões utilizadas num processo devem ser declaradas. Esta declaração é feita inserindo dentro do construtor da linguagem *Extensions* os Namespaces associados às extensões e um atributo *MustUnderstand* com o valor *yes* ou *no* que indica se o motor de execução deve compreender as extensões. Concretizar uma extensão pode ser feito em tempo de modelação ou em tempo de execução [6]. Quando realizada em tempo de modelação, uma extensão é concretizada através de uma transformação da linguagem que consiste em traduzir os elementos de extensão em elementos da norma da linguagem de tal forma que o motor de execução ignora a existência de extensões. Já em tempo de execução esta é feita alterando o motor de execução para adicionar os novos conceitos e a compreensão dos mesmos.

3.5 XSL Transformations

Extensible Stylesheet Language Transformations (XSLT) [7] é uma especificação que define a sintaxe e a semântica de uma linguagem para transformar documentos XML em outros documentos XML. XSLT é concebido para uso como parte da Extensible Stylesheet Language (XSL), que é uma linguagem de estilos para XML. XSL inclui um vocabulário XML para especificar a formatação e usa o XSLT para descrever como o documento é transformado em outro documento XML.

4 Trabalho Relacionado

No nosso trabalho utilizamos contexto com o mesmo significado que George et al. [8,9]. Estes autores descrevem contexto como um estado do ambiente, o qual é externo ao processo, cujo valor é alterado de forma independente do ciclo de vida do processo e cujo valor pode influenciar a execução do processo.

Tradicionalmente, a informação de contexto é obtida de acordo com um paradigma síncrono de pedido/resposta em determinados pontos dos processos de negócio. Esta informação pode ser utilizada para, por exemplo, determinar os serviços que compõem os processos [10], selecionar, de entre várias, a implementação de um serviço específico [11] ou determinar se um serviço deve fazer parte de uma composição [12]. No entanto, o foco do nosso trabalho é simplificar a utilização de informação de contexto nos processos de negócio.

Em [13], os autores propõem uma extensão à WS-BPEL, designada por Context4BPEL, com o objetivo de modelar explicitamente a forma como o contexto influência os fluxos de trabalho. A Context4BPEL é definida de acordo com os mecanismos de extensão da norma WS-BPEL. Esta extensão inclui mecanismos de modo a: (1) gerir eventos de contexto, ou seja, permitir a receção assíncrona de eventos; (2) interrogar dados do contexto e (3) avaliar condições de transição baseadas em dados de contexto. No entanto, a Context4BPEL estende ambos os ambientes de desenho e de execução e a gestão da informação de contexto está dependente da plataforma Nexus.

Em [14], os autores propõem uma extensão ao WS-BPEL que inclui variáveis passadas por referências. Com este tipo de variáveis, os serviços podem trocar entre si apontadores para variáveis em vez dos seus valores. Os apontadores são representados através de EPRs [3]. De acordo com o valor de um dos atributos da extensão, as referências são avaliadas (1) aquando da ativação do *Scope*, (2) antes de as variáveis serem usadas, (3) periodicamente ou (4) através de um evento enviado de um serviço exterior. A definição do processo é transformada em WS-BPEL normalizado em tempo de desenho, substituindo as referências por variáveis WS-BPEL, inserindo as ligações aos parceiros e as atividades de interação. O tipo (4) de avaliação de referências é semelhante ao objetivo proposto no nosso trabalho: a transformação adiciona um construtor *onEvent* à definição de processo. No entanto, estes autores não indicam como é que o serviço externo endereça o evento ao serviço web correspondente ao *onEvent*. Adicionalmente, a avaliação de referências está dependente do serviço RRS (Reference Resolution Service), um serviço específico da plataforma proposta por estes autores.

George et al. [8,9] propõem também uma solução baseada em variáveis de contexto. Estes autores estendem a WS-BPEL adicionando novos atributos às variáveis. No entanto, a definição do processo também tem de conter explicitamente a operação de Invoke (itálico) para efectuar a subscrição. A concretização da extensão é efectuada através da alteração do motor de execução e a distinção das instâncias de processos é efectuada através da plataforma Apache Muse [15].

5 WS-BPEL com Variáveis de Contexto

Nesta secção é descrita a nossa solução. Inicialmente é definida a extensão à linguagem de modo a incorporar o conceito de variáveis de contexto. De seguida é descrita a concretização da extensão e é apresentado o protótipo desenvolvido.

5.1 Definição da Extensão da WS-BPEL

O nosso objetivo principal é simplificar o acesso a informação de contexto dentro de um processo BPEL através do novo conceito de variável de contexto. Desta forma, cada variável representará o valor atual lido num determinado sensor. Como a linguagem já fornece um construtor específico para as variáveis, decidiu-se que a extensão seria feita adicionando novos atributos ao construtor. Os atributos escolhidos representam a informação mínima necessária para subscrever segundo a norma WS-Notifications e, por sua vez, a informação mínima para identificar um sensor num serviço web.

De seguida está um exemplo que mostra como definir uma variável de contexto chamada “tempVar” de um sensor disponibilizado num serviço web cujo endereço é representado pelo atributo publisherEPR com o tópico ”Temperatura”.

Exemplo da declaração de uma variável de contexto

```
<variables>
...
<variable name="tempVar" Type="xsd:anyURI" iotx:topic="Temperatura"
iotx:publisherEPR="http://192.168.1.43:8081/publisherService" />
</variables>
```

Adicionalmente, a extensão teve ser declarada no processo, como tal de seguida é apresentada a declaração da extensão, do seu Namespace e respetivo prefixo.

Exemplo da declaração da extensão e respetivo Namespace

```
<bpel:process name="myProcess"
...
xmlns:iotx="http://iot.extensions">
<extensions>
<extension namespace="http://iot.extensions"
mustUnderstand="yes"/>
</extensions>
...
</bpel:process>
```

Na secção seguinte apresentamos a forma como foi concretizada a extensão.

5.2 Concretização da Extensão WS-BPEL

Como referido anteriormente, as extensões WS-BPEL podem ser concretizadas em tempo de modelação ou em tempo de execução. Neste trabalho optámos por concretizar a extensão proposta em tempo de modelação, tornando-a independente do motor de execução. No entanto, ao transformar o processo são adicionadas novas atividades, variáveis, etc; que não representam exactamente o processo desenhado pelo modelador.

De seguida detalhamos a transformação efectuada, na qual a comunicação entre as instâncias dos processos e os sensores é efetuada de acordo com a norma WS-notifications. Esta transformação inclui a edição do ficheiro WS-BPEL e a criação do ficheiro WSDL.

Edição do ficheiro WS-BPEL A transformação no documento WS-BPEL ocorre quando são detectadas declarações de variáveis de contexto dentro de um processo. Quando isto acontece inicialmente são copiadas todas as variáveis, *PartnerLinks*, *CorrelationSets* e outros elementos que possam estar declarados pelo programador. Às variáveis de contexto que são encontradas são-lhes removidos os atributos de extensão passando estas a ser variáveis normais do tipo xsd:AnyURI. De seguida a *Sequence* principal do processo é mudada de forma a que a primeira atividade (a atividade *Receive* chamada "Start") continue a ser a atividade inicial. A seguir a esse *Receive* é inserido uma atividade de *Flow* que

permite que as sequências de atividades onde se recebem as notificações sejam executadas em paralelo com a sequência do programador. Dentro deste *Flow* é criada uma *Sequence* onde é colocado o fluxo de negócio do programador e, para cada variável de contexto, é adicionada ao processo uma sequência de actividades (*Sequence*). Esta sequência incluí as seguintes operações:

1. **Operação de Subscrição** – Esta operação é efectuada adicionando uma atividade *Invoke*. Esta atividade *Invoke* contacta o EPR do publicador indicado na definição da variável de contexto. Como a operação de subscrição é de dois sentidos, declara-se no processo duas variáveis, a de *input* e a de *output*. A mensagem enviada para o subscritor é inicializada por uma actividade de *Assign*, declarada antes do *Invoke*, e é formatada de acordo com a norma WS-Notifications. Nesta inicialização o tópico declarado na variável de contexto e o EPR para onde devem ser enviadas as notificações (que é gerado concatenando o nome do processo com o nome da variável de contexto) são inseridos na mensagem de subscrição. Já a variável de *output* é inicializada na resposta do *Invoke*. Como é na resposta de subscrição que se inicializa o *Correlation* (abordado posteriormente) declara-se a inicialização no sentido da resposta. Abaixo é mostrado um exemplo da mensagem de subscrição e o código da atividade *Invoke*. Esta atividade tem de ter um *PartnerLink* para usar o *PartnerRole* na invocação e como tal também é acrescentado um *PartnerLink*.

   Exemplo da mensagem de Subscrição (Subscribe)

```
<wsnt:Subscribe ...>
  <wsnt:ConsumerReference>
   <wsa:Address ... >
      http://192.168.1.71:8080/ode/processes/myProcesstempVar
    </wsa:Address>
  </wsnt:ConsumerReference>
  <wsnt:Filter>
    <wsnt:TopicExpression ... > Temperatura </wsnt:TopicExpression>
  </wsnt:Filter>
</wsnt:Subscribe>
```

   Invoke que efetua a Subscrição

```
<bpel:invoke name="Invoke" partnerLink="pubSubPartnerLink" operation="Subscribe"
 portType="wsntw:NotificationProducer"
 inputVariable="subscribeRequest"
 outputVariable="subscribeResponse">
  <bpel:correlations>
    <bpel:correlation set="notifyCorrelationSet" initiate="yes" pattern="response" />
  </bpel:correlations>
</bpel:invoke>
```

2. **Operação de recepção de notificações** - Para o processo receber as notificações é adicionado à sequência uma atividade de *Receive*(ou seja, uma atividade de comunicação de *input*). Esta atividade vai utilizar o mesmo *PartnerLink* da actividade da operação anterior mas neste caso é usado o *MyRole*. Associada a esta actividade está a declaração da variável de mensagem de notificação para onde vão ser guardadas as actualizações recebidas. Neste *Receive* está também associado o uso do *Correlation*. De seguida está um exemplo deste *Receive*.

   Receive onde chegam as notificações

```
<bpel:receive name="Receive" partnerLink="pubSubPartnerLink" operation="Notify"
portType="wsntw:NotificationConsumer" variable="NotificationMsg">
  <bpel:correlations>
    <bpel:correlation set="notifyCorrelationSet" initiate="no" />
  </bpel:correlations>
</bpel:receive>
```

   Por fim, logo a seguir a este Receive é adicionada uma actividade *Assign*, a qual copia o elemento da mensagem de notificação que contém o valor novo para a variável de contexto. Desta forma o valor actual lido do sensor fica guardado na variável. Em baixo está o código para este *Assign*.

   Assign que atualiza o valor da variável

```
<bpel:assign validate="no" name="updateVar">
  <bpel:copy>
    <bpel:from part="Notify" variable="NotificationMsg">
      <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
         wsnt:NotificationMessage/wsnt:Message
      </bpel:query>
    </bpel:from>
    <bpel:to variable="tempVar" />
  </bpel:copy>
</bpel:assign>
```

Tendo em conta que todas as instâncias usam a mesma porta para receber as notificações temos de usar o *Correlation* de forma a garantir a entrega das notificações às instâncias corretas. De seguida é explicado como usamos o *Correlation* para garantir que as notificações publicadas são entregues ao subscritor correto.

Como cada variável de contexto corresponde a uma subscrição diferente, é criado um *correlationSet* por variável. Como pretendemos usar o *Correlation* com duas mensagens (*SubscriptionResponse* e *Notify*) são definidas duas regras que serão usadas por todos os correlationSets criados pela extensão. As regras indicam, para cada mensagem, o campo *ReferenceParameters* do elemento *SubscriptionReference* como sendo para usar na correlação. Já o tipo de dados a ser

utilizado tem de ser o mesmo que o campo *ReferenceParameters*, ou seja, *anyURI* (qualquer tipo). Por fim, este *Correlation* é associado à resposta do *Invoke* que efetua a subscrição (sendo este quem inicia o *correlationSet*) e à atividade de *input* que recebe as notificações. Desta forma, quando uma mensagem de notificação chegar a um EPR de um processo garante-se que a mensagem é entregue a essa instância correta.

Adição ficheiros WSDL Como visto anteriormente, para cada variável de contexto declarada são referidos dois serviços: o serviço do publicador onde vai ser invocada a operação de subscrição e o serviço para onde vão ser enviadas as notificações. Estes serviços têm de ser declarados num ficheiro WSDL. Como o ficheiro WSDL do utilizador não deve ser alterado, decidiu-se criar um ficheiro adicional e o processo BPEL transformado importa-o. O endereço do serviço do publicador é obtido diretamente da definição da variável de contexto. Já o endereço do serviço para onde as notificações são enviadas é criado através da concatenação do nome do processo com o nome da variável de contexto. As operações de cada serviço são diretamente importadas das interfaces de publicador (*NotificationProducer*) e subscritor (*NotificationConsumer*), disponibilizadas pelos WSDL da norma WS-Notifications. Para além dos serviços, é neste documento que são declaradas as propriedades de *Correlation* usadas na transformação BPEL.

5.3 Protótipo

O protótipo foi desenvolvido com as seguintes ferramentas:

- Apache Tomcat [16]
- Apache ODE – Motor WS-BPEL [17]
- Saxon Home Edition – Processador XSLT [18]
- Eclipse EE + BPEL Designer plugin [19,20]

O nosso protótipo consiste em três componentes:

1) O processador XSLT para executar a transformação; 2) O motor de execução WS-BPEL onde vão ser executados os processos BPEL e 3) O serviço web que disponibiliza os sensores que desejamos associar às variáveis de contexto. O motor XSLT escolhido, o Saxon-HE9.4, foi selecionado devido ao facto de suportar XSLT 2.0. A versão 2.0 consegue, entre outras coisas, gerar vários ficheiros de *output*, que era uma das necessidades da nossa extensão. Com recurso a este processador foi desenvolvido um script que gera o *output* explicado anteriormente. A figura 1 ilustra como a extensão é efetuada em tempo de modelação usando as ferramentas indicadas anteriormente.

Como o protótipo foi implementado para ser usado no Apache ODE, foi também incluído na extensão um ficheiro deploy.xml parcialmente preenchido. Este ficheiro é usado pelo ODE para fazer o mapeamento entre os *PartnerLinks* e os serviços declarados. A adição deste ficheiro serve unicamente para poupar ao utilizador o tempo de preencher todos os serviços criados pela transformação.

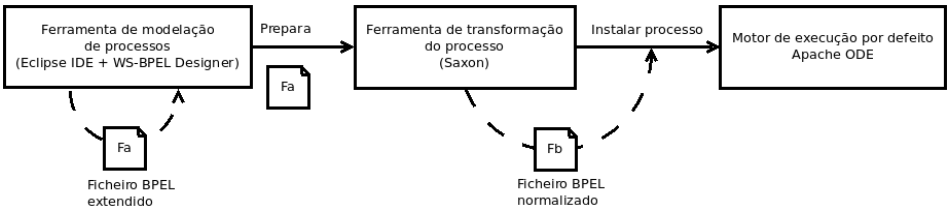

Figura 1. Funcionamento da extensão em tempo de modelação

6 Conclusões e Trabalho Futuro

Os processos de negócio podem beneficiar de forma significativa da informação da IoT. O trabalho apresentado neste artigo pretende simplificar o acesso a esta informação pelos processos WS-BPEL. Através de uma extensão à WS-BPEL, os processos podem incluir variáveis de contexto, cujo valor é actualizado de forma assíncrona. No entanto, as operações necessárias para se efectuar a comunicação entre a instância do processo e os sensores são da responsabilidade da extensão, permitindo ao modelador do processo focar-se na lógica do negócio. A extensão proposta é concretizada em tempo de modelação, permitindo que seja utilizada em qualquer motor de execução.

O trabalho futuro será efectuado em quatro linhas. Como o *Correlation*, atualmente, só usa um elemento do EPR, pretende-se adicionar regras para os restantes elementos. Pretende-se melhorar alguns aspectos da transformação de modo a incluir mais validações e a contemplar variáveis de contexto declaradas dentro de *Scopes*. Pretende-se também criar uma extensão para o plugin WS-BPEL Designer do Eclipse, ferramenta utilizada no protótipo para a modelação de processos, para suportar as variáveis de contexto. Finalmente, tendo por base as variáveis de contexto, pretende-se suportar processos mais reactivos, adicionando novos construtores à linguagem, como por exemplo, o *when-then* e suportando alterações ad-hoc às instâncias dos processos.

Referências

1. Web Services Business Process Execution Language Version 2.0 - OASIS Standard, Abril 2007.
2. Deze Zeng, Song Guo, Zixue Cheng. The Web of Things: A Survey School of Computer Science and Engineering, The University of Aizu, Japan. 2011 [http://ojs.academypublisher.com/index.php/jcm/article/view/jcm0606424438/3601]
3. Web services addressing (WS-addressing) - IBM, W3C 2004 http://www.w3.org/Submission/ws-addressing/
4. Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. Web Service Definition Language. Microsoft, IBM. 2001 http://www.w3.org/TR/wsdl
5. Web services notification (WS-notification) version 1.3. - OASIS Standard 2006 https://www.oasis-open.org/committees/wsn/

6. Oliver Kopp, Katharina Görlach, Dimka Karastoyanova, Frank Leymann, Michael Reiter, David Schumm, Mirko Sonntag, Steve Strauch, Tobias Unger, Matthias Wieland, Rania Khalaf. A Classification of BPEL Extensions. Institute of Architecture of Application Systems University of Stuttgart, IBM TJ Watson Research Center. 2011 ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/ART-2011-18/ART-2011-18.pdf
7. James Clark, Stephen Deach, Michael Kay. XSL Transformations. Saxonica, Adobe. 2007 http://www.w3.org/TR/xslt20/
8. Allen George, Paul Ward. An architecture for providing context in WS-BPEL processes. In Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds (CASCON '08), Marsha Chechik, Mark Vigder, and Darlene Stewart (Eds.). ACM, New York, NY, USA. 2008 http://doi.acm.org/10.1145/1463788.1463818
9. Allen George. Providing Context in WS-BPEL Processes. Journal of the Electrochemical Society. 2008. http://libdspace.uwaterloo.ca/handle/10012/4025
10. Lian Yu and Shuang Su. Adopting context awareness in service composition. In Proceedings of the First Asia-Pacific Symposium on Internetware (Internetware '09). ACM, New York, NY, USA. 2009 http://doi.acm.org/10.1145/1640206.1640217
11. Anand Ranganathan, Scott McFaddin. Using Workflows to Coordinate Web Services in Pervasive Computing Environments. In Proceedings of the IEEE International Conference on Web Services (ICWS '04). IEEE Computer Society. 2004 http://dx.doi.org/10.1109/ICWS.2004.119
12. Dimka Karastoyanova, Alejandro Houspanossian, Mariano Cilia, Frank Leymann, Alejandro Buchmann. Extending BPEL for Run Time Adaptability. In Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference (EDOC '05). IEEE Computer Society. 2005 http://dx.doi.org/10.1109/EDOC.2005.14
13. Matthias Wieland and Oliver Kopp, Daniela Nicklas, Frank Leymann. "Towards Context-aware Workflows". In: Pernici, Barbara (ed.); Gulla, Jon Atle (ed.): CAiSE'07 Proceedings of the Workshops and Doctoral Consortium Vol.2, Junho 11-15º, 2007
14. Wieland, M.; Gorlach, K.; Schumm, D.; Leymann, F. . Towards Reference Passing in Web Service and Workflow-Based Applications. Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE International. 2009 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5277706&isnumber=5277609
15. Apache Muse - http://ws.apache.org/muse/ 14 Junho, 2012
16. Apache TomCat http://tomcat.apache.org/ 14 Junho, 2012
17. Apache ODE http://ode.apache.org/ 14 Junho, 2012
18. Saxon Home Edition http://saxon.sourceforge.net/ 14 Junho, 2012
19. Eclipse IDE for Java EE Developers http://www.eclipse.org/ 14 Junho, 2012
20. BPEL Designer Project http://www.eclipse.org/bpel/ 14 Junho, 2012