Reconfiguração Dinâmica Estruturada de Workflows de Serviços Web

Filipe Araújo, Maria Cecília Gomes e Hervé Paulino

CITI / Departamento de Informática
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
2829-516 Caparica, Portugal

Resumo O conceito de serviço consiste numa abstracção simples mas poderosa para representar/integrar entidades de diferentes contextos, e.g. serviços Web da área de negócio, serviços computacionais em Grid, ou *Internet of Things*. Os sistemas de gestão de*workflows*, por seu lado, são uma solução standard para a especificação/execução da composição de serviços, para os quais é necessário garantir mecanismos adicionais de reconfiguração dinâmica. Estes resultam de novos requisitos e.g. coordenação de serviços representando entidades bem distintas e/ou formando aplicações de larga escala, ou de requisitos aplicacionais que é necessário incorporar em tempo de execução do workflow. Neste trabalho propomos uma extensão com mecanismos de reconfiguração dinâmica baseados no conceito de padrão, de uma ferramenta de *workflow* da área de negócio, i.e. oferecendo orquestração de serviços segundo um modelo de *fluxo de controlo*. Os padrões implementados seguem o modelo de *fluxo de dados* característico da área científica/sistemas de eventos. As reconfigurações dinâmicas são bem definidas e confinadas, dado que se restringem a estes padrões.

Keywords: Serviços *Web*, Sistemas de *Workflow* Adaptáveis, Padrões de desenho.

1 Introdução

A computação orientada a serviços emergiu como um paradigma consensual para a abstracção de diferentes tipos recursos, com ou sem estado, tendo a simplicidade do conceito de serviço desencadeado uma tendência de disponibilizar *tudo como um serviço* (*XaaS* – *everything as a service*). Os exemplos vão desde: a) a área de *redes de sensores sem fios* (WSN), para disponibilizar uma interface de mais alto nível simplificando a sua parametrização e recolha de dados [15]; b) o domínio da *Internet das Coisas* (*Internet of Things* – *IoT*) [6] para integrar diferentes tipos de entidades e objectos quer reais quer virtuais; até c) serviços no contexto de computação em *Cloud* (e.g. *IaaS*, *PaaS*, ou *SaaS*) [14].

O sucesso do modelo de serviços resulta também da simplicidade do modelo de interacção pedido/resposta sem estado presente nos serviços *web* na sua forma primitiva, que permite um desacoplamento entre o serviço e o seu acesso

por parte de vários clientes. No entanto, a utilização da noção de serviço como interface para o acesso a actividades continuadas no tempo e recursos com estado, tem merecido uma atenção acrescida sendo inclusivamente usada nos exemplos referidos. Por exemplo, a submissão de tarefas de longa duração, como é o caso de alguns serviços de *Cloud* e *Grid computing*, requer que os clientes possam interrogar o serviço sobre o estado dessas tarefas. Na IoT é comum ter entidades representadas através de serviços com estado, dado que muitas destas pertencem a um único dono (e.g. dispositivos electrónicos numa casa) não sendo necessário contemplar o acesso ao serviço por diferentes utilizadores. Já o acesso a WSNs requer serviços que permitam a obtenção dos dados recolhidos pelas redes sem que os utilizadores tenham de estar continuamente a interrogar o serviço. Este tipo de serviços requer assim realização de um estado dinâmico que tem de ser mantido consistente ao longo da troca de várias mensagens entre o serviço e cada um dos clientes [3]. Tal tem potenciado o surgimento de modelos de interacção mais complexos para além do tradicional acesso pedido/resposta, e.g. subscrição de eventos ou disseminação de dados baseada em *stream*.

A crescente complexidade destas arquitecturas orientadas a serviços tem obrigado à disponibilização de mecanismos para a sua especificação/composição e que garantam novas formas de automatismo e dinamismo na sua execução. Neste contexto, surge o conceito de *workflow* (fluxo de trabalho) de serviços *web* representando as interdependências entre serviços, e oferecendo o automatismo necessário à sua interacção e interoperabilidade.

Mecanismos de Adaptação em Sistemas de Workflow de Serviços

Nos sistemas *workflow* de serviços *web* é necessário garantir mecanismos de adaptação dinâmica no suporte ao acesso, interligação, e coordenação de serviços. Tal como descrito em [8], estes mecanismos são necessários tanto para os workflows da área de negócio como da área científica, e.g. tratamento de excepções respondendo a alterações no ambiente resultando na substituição de um serviço por outro mais eficiente ou fiável; alteração do workflow de modo a incorporar novos requisitos ao nível da aplicação.

Por sua vez, a interacção com serviços que representam recursos com estado ou actividades de longa duração, requer modelos de interacção mais complexos e mecanismos de suporte à sua adaptação dinâmica [7]. Assim, pode ser benéfico mudar o modelo de interacção usado quer como resultado de alterações no contexto dos serviços ou dos dados em trânsito, quer dos seus clientes, ou da interacção entre ambos. Por exemplo, reduzir a cadência da entrega de dados num *stream* caso o receptor não tenha capacidade de realizar o processamento desses dados à mesma velocidade; ou a detecção de valores elevados de temperatura em WSNs numa determinada zona resultando na mudança do modelo de interacção publicador/subscritor para streaming para a obtenção contínua desses valores.

De igual modo, a disponibilização de modelos dinâmicos de interacção segundo o modelo de fluxo de dados para estruturar/coordenar um conjunto de tarefas num sistema de workflow da área de negócio permite, por exemplo, a) representar aplicações da área de negócio que beneficiem desses modelos dinâmicos,

e.g. poder coordenar tarefas no workflow representando actividades de humanos cuja execução é desencadeada por eventos gerados no contexto do modelo publicador/subscritor; e ainda poder alterar dinamicamente este modelo de interacção para produtor/consumidor quando um conjunto de dados produzidos não pode ser perdido; b) representar aplicações na área de ciência/engenharia, onde um stream de dados flui por um conjunto de tarefas em execução concorrente, estando estas incluídas no contexto de um workflow regido pelo modelo de fluxo de controlo; e ainda poder reconfigurar dinamicamente esse conjunto de tarefas, e.g. adicionar novas tarefas que consomem esse stream de dados; c) poder propagar os dados gerados por um serviço que faz a interface com recursos com estado, ao longo das tarefas do workflow e com esse mesmo modelo de interacção, e.g. uma tarefa do workflow recebe um stream de dados de um desses serviços, sendo esse stream propagado a um conjunto de outras tarefas no workflow; e ainda caso o modelo de interacção com esse serviço seja dinamicamente alterado para publicador/subscritor, alterar também dinamicamente o modelo de interacção entre a tarefa que acede a esse serviço com as outras tarefas no workflow. O trabalho proposto neste artigo tem precisamente como objectivo disponibilizar este tipo de mecanismos de reconfiguração dinâmica, bem como contribuir para o suporte à definição estruturada de *workflows* tanto das áreas de ciência/engenharia como de negócio.

A próxima secção descreve a solução proposta e a secção 3 descreve a extensão da ferramenta de workflow escolhida. A secção 4 apresenta um exemplo de aplicação e a 5 faz a comparação com alguns trabalhos relacionados. Finalmente, as conclusões e trabalho futuro são apresentados em 6.

2 Reconfiguração Dinâmica de Workflows com base em Padrões

A solução proposta neste trabalho baseia-se no conceito de padrão, visto que os padrões são aplicados como formas estruturadas de interacção entre tarefas num *workflow*, e servem de base aos mecanismos de reconfiguração dinâmica suportados. Os padrões são disponibilizados como entidades de primeiro nível com o seu próprio ciclo de vida (ver [11]), permitindo que sejam manipulados individualmente bem como combinados com outros conceitos de primeiro nível (e.g. serviços ou componentes) num *workflow*. Os padrões são disponibilizados como templates parametrizáveis nas dimensões de estrutura e comportamento, permitindo a a sua combinação flexível (incluindo a construção de hierarquias) e a sua reconfiguração individual (também nas dimensões de estrutura e comportamento).

Os templates de padrões de estrutura implementados foram topologias comuns como estrela, anel, e *pipeline*, podendo as topologias em árvore ser implementadas através de hierarquias destas topologias. Os elementos desses padrões de estrutura podem ser anotados com papéis de um (único) padrão de comportamento, tendo sido implementadas versões simplificadas de produtor/consumidor, publicador/consumidor e streaming, estando o modelo cliente/servidor

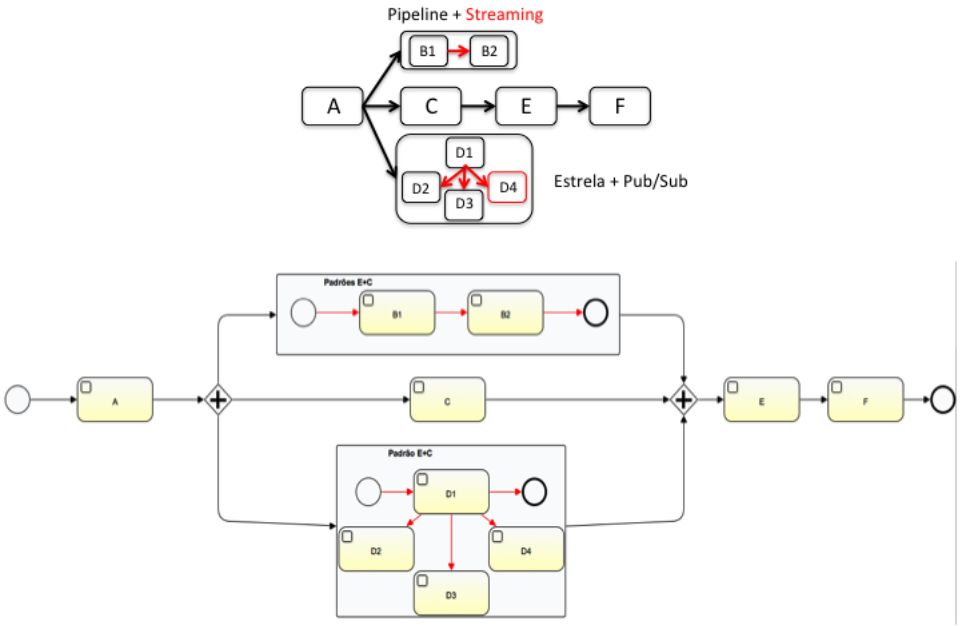


Figura 1: *Workflow* abstracto com dependências de fluxo de controlo e de fluxo de dados, no contexto de padrões de estrutura.

disponível por defeito no acesso aos serviços. As reconfigurações podem ser realizadas em tempo estático e dinâmico, e são conformes/obedecem à semântica dos padrões manipulados. Por exemplo, qualquer alteração a um *pipeline* tem de resultar num *pipeline* (e.g. incrementar o seu número de etapas). Por simplificação, a substituição de um comportamento por outro no contexto de um padrão estrutural afecta de igual modo todos os elementos do mesmo tipo (e.g. passar de publicador/subscritor para produtor/consumidor numa estrela implica que todos os satélites passam de subscritores a consumidores).

A notação usada neste trabalho para representar os modelos de fluxo e de dados, está ilustrada na Figura 1: as dependências no *workflow* geral, no topo da Figura, são de fluxo de controlo (a preto); e as dependências em cada padrão são de fluxo de dados (a vermelho), i.e. correspondem às dependências definidas de acordo com o padrão de comportamento usado no contexto desses padrões de estrutura; na parte inferior da Figura 1 é apresentado o *workflow* correspondente no contexto da ferramenta de *workflow Activiti* [1] escolhida para implementar este trabalho. Compreendemos que a notação usada possa não ser a mais clara, nem tem correspondência na notação *Business Process Model and Notation (BPMN)* [2], situação esta que pretendemos corrigir em trabalho futuro. Segue-se a descrição da ferramenta Activiti.

3 Ferramenta Activiti Estendida

A escolha da ferramenta Activiti [1] resultou da análise de um conjunto de ferramentas da área de negócio considerando os requisitos descritos na secção 5. O seu

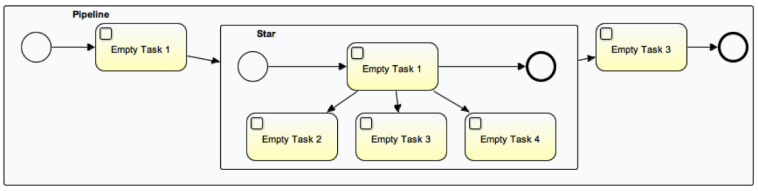


Figura 2: Hierarquia de topologias

principal objectivo é especificar processos na linguagem *BPMN 2.0*, facilitando a colaboração de pessoas da área de negócio e responsáveis de desenvolvimento através de uma comunidade de partilha de documentação e desenvolvimentos. Esta ferramenta disponibiliza um número vasto de componentes e tem como base um motor de execução de processos *BPMN 2.0*. Estando consolidada em várias empresas, a ferramenta tem sofrido constantes evoluções nas suas funcionalidades. Embora seja direccionada para desenvolvimentos em linguagem *Java*, a ferramenta tem a particularidade de partilhar o código fonte e ser distribuída sobre licença *Apache*[1]. A *Activiti* contém várias componentes que podem ser integradas em qualquer aplicação *Java*, daí poder ser instalada num servidor, num *cluster* ou até mesmo em serviços *cloud*. A arquitectura do protótipo implementado é suportada por duas das componentes oferecidas – *Activiti Designer* (um *plugin* Eclipse) e *Activiti Modeler*, permitindo a definição de *workflows* que incluem tarefas que acedem a serviços *web* denominadas *service tasks*.

Com o objectivo de disponibilizar uma ferramenta na área de negócio suportando modelos dinâmicos do tipo fluxo de dados (*dataflow*), a ferramenta Activiti foi estendida com padrões de comportamento aplicados no contexto de templates de padrões de estrutura parametrizáveis. Na sua implementação tentou-se utilizar ao máximo as características oferecidas pela notação *BPMN 2.0* e o suporte à sua execução no contexto da Activiti. Por um lado, o projecto *Activiti Designer (Eclipse Plugin)* foi estendido com quatro tipos de padrões de estrutura (topologias) como tarefas abstractas – *pipeline*, estrela, anel e *composite* (e.g. um exemplo de um pipeline e de uma estrela está na parte inferior da Figura 1). A extensão foi feita com a sua adição dos padrões à *palete* de elementos do plugin, bastando fazer *drag and drop* para o ambiente de composição. A plataforma suporta também hierarquias de estruturas, como seja a inclusão de uma estrela na segunda etapa de um *pipeline*, ilustrada na Figura 2.

A implementação dos padrões de comportamentos sobre essas topologias foi feita com base em padrões de fluxo de controlo em *workflows* [4] – *loop*, sincronização, junção simples e divisão paralela. Estes são os blocos de construção de dois *workflows* sendo subjacentes à execução dos comportamentos publicador/subscritor (pub/sub), produtor/consumidor (prod/cons) e *streaming*, bem como aos mecanismos que permitem a sua reconfiguração dinâmica. Como tanto a interacção entre tarefas como a reconfiguração dinâmica têm requisitos de adaptação que não existem na ferramenta Activiti, foram desenvolvidas as duas tarefas apresentadas na Tabela 1.

[1] Apache License, Version 2.0. http://www.apache.org/licenses/LICENSE-2.0

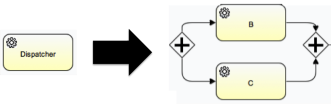
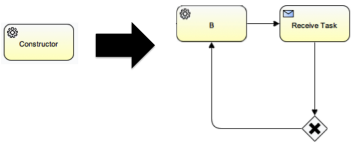
Nome tarefa	Descrição	Exemplo	Descrição exemplo
Dispatcher	Tarefa responsável por, em tempo de execução, construir um *workflow* a partir de um grupo de tarefas. Pode ser aplicada várias vezes ao longo da execução.		Dadas as tarefas A e B, é construído o *workflow* apresentado, que executa ambas as tarefas em paralelo.
Constructor	Tarefa responsável por, em tempo de execução, construir estruturas comuns em *workflows*. A sua execução transforma-a numa estrutura, o que implica que só pode ser usada uma única vez.		A figura representa uma estrutura típica em *workflows*, *loops*. Com o *gateway* exclusivo o ciclo é mantido, até a condição associada for falsa.

Tabela 1: Tarefa *Dispatcher* e *Constructor*

Dos dois *workflows* construídos, o primeiro serve de base à implementação do padrão pub/sub. A publicação de um valor pode ter como consequência a execução de uma ou mais tarefas, consoante o número de subscrições interessadas no valor em causa. Este grau de dinamismo é suportado pela tarefa *Dispatcher* que se converte num *workflow* por forma executar, em paralelo, todas as tarefas subscritoras geradas. Este *workflow* é composto por um ciclo e um *gateway* paralelo, suportando a co-existência de múltiplas instâncias para o atendimento simultâneo de vários valores produzidos. Para implementar os comandos que desencadeiam reconfigurações dinâmicas através da consola de testes oferecida pela componente *Activiti Designer (Eclipse Plugin)*, é usada a tarefa *Reconfigurator* que interpreta os comandos inseridos; consoante o comando inserido, é feita uma transformação do *workflow*, e.g. pub/sub para prod/cons. Este *workflow* de atendimento do comportamento prod/cons contém a tarefa *Constructor*, que é utilizada para simplificar os casos em que é necessário aplicar uma reconfiguração isolada no tempo, como é o caso deste exemplo que transforma o *workflow* num ciclo de atendimento dos consumidores.

Salienta-se que o comportamento prod/cons implementado é uma versão simplificada da definição geral usada no contexto de uma sessão em [12]: todos os clientes de uma sessão com modelo de interacção prod/cons são consumidores dos dados; do mesmo modo, todos as tarefas satélite num padrão estrela consomem os mesmos dados, e que são produzidos pela tarefa que instância o núcleo dessa estrela. No entanto, sendo estes consumidores executados em paralelo (cada um suportado por uma thread), garante-se que existe independência entre o rácio de consumo e de produção, como é devido num paradigma prod/cons. Este padrão também é reconfigurável dinamicamente, existindo também uma tarefa *reconfigurator* para o *workflow* de suporte à implementação deste comportamento.

4 Exemplo de Aplicação

Este exemplo ilustra uma possível aplicação do nosso protótipo no contexto de sistemas aplicacionais orientados a dados dinâmicos *(Dynamic data driven applications systems (DDDAS))*[9]. Em geral, o conceito *DDDAS* captura a necessidade de incorporar dados dinamicamente numa aplicação de simulação em execução, bem como a possibilidade de a própria aplicação parametrizar o modo como essa recolha de dados é feita. O exemplo escolhido é um caso particular de uma simulação no contexto de monitorização e análise de cheias urbanas (*Urban Flooding Analysis and Monitoring (UFAM))*, descrevendo alguns cenários possíveis para uma zona crítica que, recorrentemente, é sujeita a situações de cheias. A aplicação está sob controlo de uma autoridade local, responsável por monitorizar e agir em situações de emergência (cenários extraídos de[2]). Para um sistema *UFAM* é necessário obter dados estatísticos de informação meteorológica (e.g. humidade ou precipitação) por forma a prever a probabilidade de cheias em locais historicamente mais afectados. Esses dados podem ser subscritos pela aplicação a partir de serviços e, usando um modelo de interacção que pode variar consoante se trata de uma situação normal, de prevenção, ou de emergência. Para mais, a aplicação é suportada por um *workflow* que permite reconfigurações dinâmicas de acordo com o evoluir dessas situações. O *workflow* base é composto por um padrão estrutural em estrela, em que o nó núcleo representa a tarefa responsável por obtenção de valores meteorológicos da zona crítica (e.g. humidade); os nós satélites representam as tarefas que utilizam esses valores e, para objectivos diferentes (e.g. disponibilizando a informação numa *interface* ou salvaguardando-a numa base de dados). O acesso às fontes de dados é simulado através de serviços *web* de testes que geram valores para os diferentes tipos de dados, e de acordo com o cenário em questão.

4.1 Situação de Prevenção

Numa situação normal, apenas são subscritos valores de humidade com o modelo de interacção publicador/subscritor, os quais são enviados para uma tarefa (satélite) que os guarda num repositório. No entanto, assim que os valores notificados excedem um valor limite pré-definido, o sistema é reconfigurado dinamicamente passando para um cenário de prevenção (Figura 3). Dinamicamente, é feita a subscrição de dados de precipitação e é adicionada uma nova tarefa (satélite) correspondendo a uma *interface* de utilizador (*UI*) acessível por membros da autoridade local. Esta nova tarefa subscreve valores de precipitação superiores a 70 mm, permitindo avaliar se a situação pode evoluir para uma situação de emergência. O comando de adição da tarefa é assim invocado em tempo de execução do *workflow*, resultando na sua reconfiguração dinâmica. Após invocação do botão de *update*, aparece a estrela já reconfigurada, como ilustrado na Figura 3. Este botão encontra-se na área de parametrizações do diagrama principal, e uma classe "SubscreverPrecipitacao"incluída no comando, cria uma ligação TCP com um servidor para obtenção dos novos valores.

[2] `http://citi.di.fct.unl.pt/postgrad/postgrad.php?id=428`

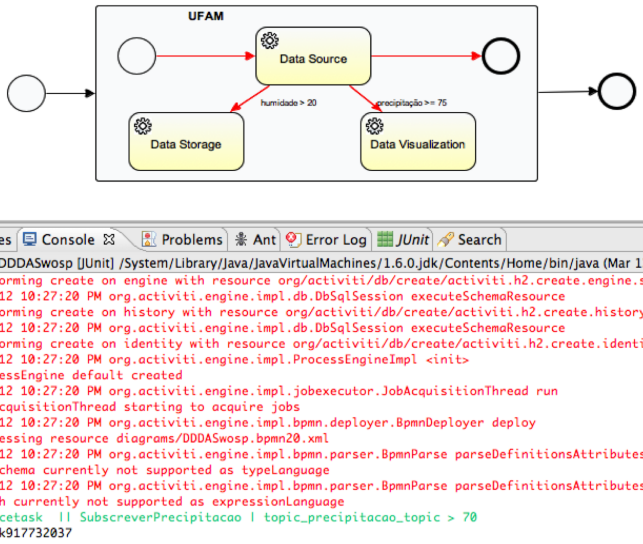


Figura 3: Exemplo *UFAM* - situação de prevenção

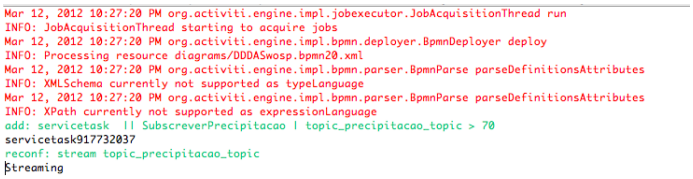


Figura 4: Exemplo *UFAM* - situação de emergência

4.2 Situação de Emergência

Caso os valores da precipitação passem a ser bastante elevados, a autoridade local necessita de receber um maior volume desses dados para um conhecimento mais preciso (situação de emergência). Requer assim a alteração do modelo de interacção para streaming (Figura 4), bem como a adição de uma nova tarefa subscrevendo a recepção de dados do nível das águas (Figura 5). Tal é feito pressionando o botão *increase*, podendo a nova tarefa ser instanciada e.g. com um novo *"Data storage"* ou com uma instância de uma consola de visualização de dados.

Supondo que ocorre um cenário de inundação, a autoridade local pode reconfigurar o *workflow* de modo a combinar/filtrar os dados recolhidos no terreno, com informação geográfica na zona, bem como lançar a execução de uma aplicação de simulação que processa esses dados. Para tal, pode definir um pipeline que consiste nestas três etapas e, por forma a guardar estes dados processados e/ou permitir a sua visualização, pode reconfigurar dinamicamente esse pipeline gerando a configuração apresentada na Figura 6. O modelo de interacção escolhido pode ser streaming (ou o produtor/consumidor, se for necessário evitar perdas de informação). O administrador, encarregue de gerir a execução deste *workflow*, pode executar o comando de adicão de mais tarefas para visualização dos dados

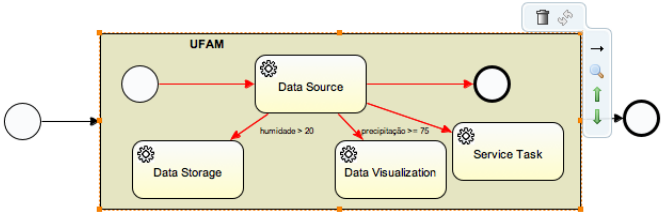


Figura 5: Exemplo *UFAM* - subscrição nível das águas

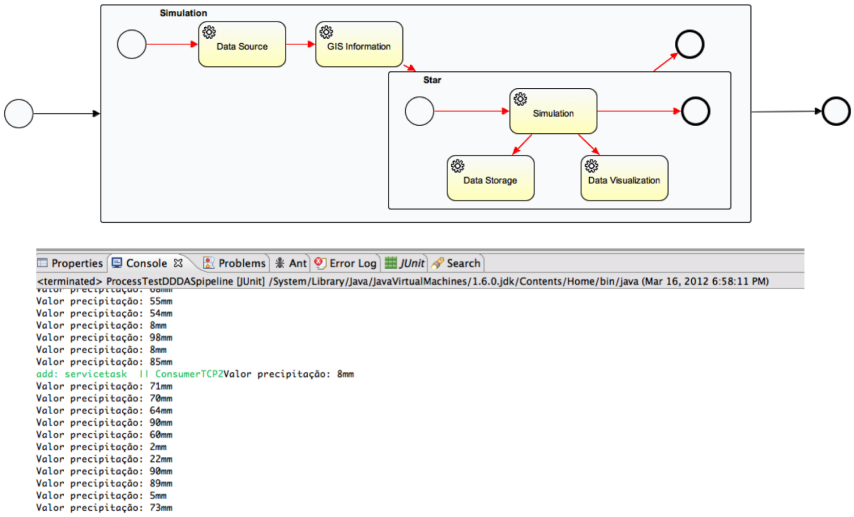


Figura 6: *Pipeline* com novo consumidor e comando de reconfiguração

numa consola permitindo que, peritos na área (e.g. em localizações geográficas diferentes) contribuam com os seus conhecimentos.

4.3 Integração Conceptual com Outros Sistemas de *Middleware*

Uma extensão que se pretende realizar é a integração deste trabalho com outros dois *middlewares*[3] que permitem o acesso a serviços representando recursos com estado, n o contexto de uma *sessão*. Este acesso é suportado por modelos de interacção dinâmicos baseados no conceito de padrão de comportamento, tendo sido disponibilizados os mesmos padrões que os aqui descritos (ver [12]). Tal permitirá o acesso a diferentes fontes de dados (e.g. recolhidas por WSNs com interfaces *web*) usando um modelo de interacção particular bem como, a propagação desses valores pelas tarefas do *workflow* usando o mesmo modelo de interacção. O objectivo é poder gerar configurações como a apresentada na Figura 7, onde uma tarefa do workflow é um dos clientes da sessão. Tal possibilita a reconfiguração dinâmica quer no contexto da sessão, quer do *workflow*.

[3] `http://citi.di.fct.unl.pt/postgrad/postgrad.php?id=377` e `http://citi.di.fct.unl.pt/postgrad/postgrad.php?id=477`

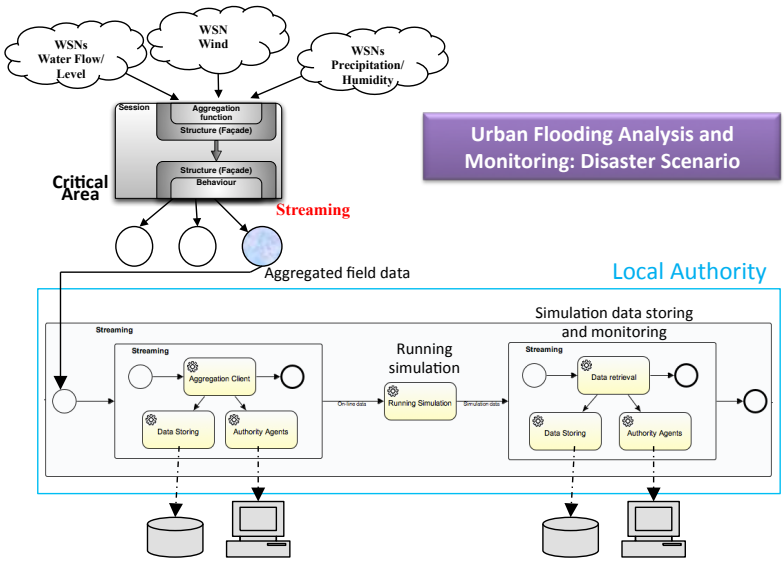


Figura 7: Situação de emergência.

Pretende-se assim fechar o ciclo de controlo característico dos sistemas DDDAS, descrito acima, em que uma simulação pode parametrizar, ou adicionar dinamicamente, as fontes de dados. Inversamente, dados recolhidos nestas fontes e, reconfigurações dinâmicas no contexto de uma sessão (e automaticamente notificadas à tarefa cliente da sessão), podem resultar em adaptações dinâmicas no *workflow*/na simulação. Considerando o exemplo descrito antes, o *workflow* ilustrado na Figura 7 permite várias reconfigurações dinâmicas: alterar/parametrizar as fontes de dados; alterar o modelo de interacção usado no contexto da sessão (e.g. *streaming na Figura*); usar esse mesmo modelo no contexto do *workflow* de modo a garantir uma propagação adequada desses dados; inversamente, reconfigurar dinamicamente o *workflow* implicando reconfigurações dinâmicas no contexto da sessão. Em todos os casos, a reconfiguração poderá ser a pedido, ou resultante de regras pré-definidas.

5 Trabalho Relacionado

Foram avaliadas outras linguagens para a composição de serviços *web*, e.g. *BPEL*, *WS-CDL*, *YAWL*, mas foi escolhida a *BPMN 2.0* dado que cumpre os seguintes requisitos a) possibilita o acesso/composição de serviços *web* fracamente acoplados num *Workflow*; b) é uma linguagem de modelação da área de negócio com ferramentas que fornecem um ambiente de desenvolvimento/execução ao utilizador, e para diferentes plataformas; c) o código fonte está disponível; e d) permite a implementação de padrões de workflow [4]. Embora existam outras ferramentas que suportam BPMN 2.0 (e.g. *jBoss*), foi escolhida como ambiente de desenvolvimento a ferramenta Activiti [1] em particular um *plugin* para o

eclipse, dado que disponibiliza um editor para a especificação de *workflows* em *BPMN 2.0* mas também um *Workflow Engine* para a sua execução. Os padrões de workflow não são directamente disponibilizados pela Activiti, mas podem ser implementados tal como descrito em [17].

Outros trabalhos referem o interesse de utilizar/estender uma ferramenta de workflow da área de negócio para suportar aplicações na área científica/de engenharia, e.g. com o objectivo de incluir tarefas manuais assíncronas e mecanismos de controlo mais sofisticados, ou de representar a geração/captura de eventos e excepções/erros. O trabalho [16] faz uma separação em níveis, permitindo que um workflow geral com dependências seguindo o modelo de fluxo de controlo (*control-flow*), orquestre o acesso a/integração com workflows da área científica, os quais apresentam dependências do tipo fluxo de dados (*dataflow*). O trabalho [10] suporta coreografias através da abstracção de *workflow skeletons* representando um conjunto de *proxies*, e que permite que os dados em trânsito entre estes não passem pela ferramenta de orquestração que gere a totalidade do *workflow*. Em termos da reconfiguração dinâmica com base no conceito de padrão, alguns sistemas usam-no para implementar mecanismos de auto-adaptação, em particular, recorrendo a padrões cuja própria arquitectura permite a sua reconfiguração [13]. Este é o caso do padrão publicador/subscritor, e.g. que permite a alteração dinâmica quer do número de publicadores quer do número de subscritores; do mestre/escravo, e.g. alterando o número de escravos de modo a melhorar o desempenho das tarefas [5]. No entanto, nenhum dos trabalhos referidos refere mecanismos de reconfiguração dinâmica como os apresentados nesta proposta.

6 Conclusões e Trabalho Futuro

Este trabalho descreve uma abordagem estruturada com base no conceito de padrão, para a reconfiguração dinâmica de *workflows* de serviços (sem estado ou representando recursos com estado). Foi escolhida uma ferramenta específica da área de negócio suportando processos BPMN 2.0 designada Activiti, e sendo uma ferramenta recente encontra-se ainda em desenvolvimento. A Activiti foi estendida com templates instanciáveis de padrões de estrutura (e.g. estrela) e comportamento (e.g. produtor/consumidor) que podem ser combinados (e.g. formando hierarquias) e, reconfigurados dinamicamente. Esta abordagem, permite definir aplicações características da área de negócio que beneficiem de modelos de interacção entre tarefas de fluxo de dados, bem como da área científica que benificiem de padrões de controlo de fluxo entre tarefas. O trabalho implementado atingiu os objectivos propostos, e segue os conceitos/requisitos identificados em [8] em termos da adaptação dinâmica de workflows, bem como técnicas aí propostas (e.g. criação de tarefas abstractas, alteração da estrutura tanto na fase de desenho como em tempo de execução).

Como trabalho futuro, pretendemos disponibilizar padrões estruturais úteis no contexto de aplicações destribuídas (e.g. *proxy* e *façade*)[11], bem como alterar a notação usada para a especificação dos padrões de comportamento, por exemplo através da especificação de coreografias em BPMN 2.0, e.g. com *pools* e *lanes*,

adaptando-as com os mecanismos de reconfiguração dinâmica referidos neste trabalho. Pretendemos ainda permitir adaptações automáticas, pré-definidas pelo utilizador, bem como tirar partido de versões mais recentes da ferramenta *Activiti* que suportem a execução concorrente de tarefas. Finalmente, será feita a integração com outros middlewares permitindo aplicações como a descrita na secção 4.3.

Agradecimentos: Este trabalho foi parcialmente financiado pelo PEst-OE/EEI/UI0527/2011 Centro de Informática e Tecnologias da Informação (CITI/FCT/UNL) - 2011-2012, e projectos MACAW – PTDC/EIA-EIA/115730/2009 e SITAN – PTDC/EIA-EIA/113729/2009.

Referências

1. Activiti BPMN platform. `http://www.activiti.org/`.
2. Business process model and notation 2.0. `http://www.bpmn.org/`.
3. Oasis web services resource framework (wsrf) tc. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf`.
4. Workflow patterns. `http://www.workflowpatterns.com/`.
5. Marco Aldinucci, Marco Danelutto, and Peter Kilpatrick. Towards hierarchical management of autonomic components: A case study. In *Proceedings of the 17th Euromicro Int. Conf., PDP 2009*. IEEE Computer Society, 2009.
6. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Comput. Netw.*, 54(15), 2010.
7. Adérito Baptista, M. Cecília Gomes, and Hervé Paulino. Session-based dynamic interaction models for stateful web services. In Mehdi Snene, editor, *IESS 2012*, number 104 in LNBIP. Springer-Verlag, 2012.
8. Manuel Caeiro-Rodriguez, Thierry Priol, and Zsolt Németh. Dynamicity in scientific workflows. Technical Report 0162, CoreGRID Netwk Excellence, 2008.
9. Frederica Darema. Dynamic data driven applications systems: A new paradigm for application simulations and measurements. In *Int. Conf. on Computational Science*, volume 3038. Springer LNCS, 2004.
10. Tino Fleuren, Joachim Gotze, and Paul Muller. Workflow skeletons: Increasing scalability of scientific workflows by combining orchestration and choreography. In G. Zavattaro et al, editor, *ECOWS*, pages 99–106. IEEE, 2011.
11. Cecilia Gomes, Omer F. Rana, and Jose Cunha. Extending grid-based workflow tools with patterns/operators. *Int.J.HPCA*, 22, 2008.
12. M. Cecília Gomes, Hervé Paulino, Adérito Sarmento Baptista, and Filipe Jorge da Silva Araújo. Dynamic interaction models for web enabled wireless sensor networks. In *MUE 2012*. IEEE CS, 07 2012.
13. Markus C. Huebscher and Julie A. McCann. A survey of autonomic computing degrees, models, and applications. *ACM Comput. Surv.*, 40, 2008.
14. Peter Mell and Tim Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6), 2009.
15. Hervé Paulino and João Ruivo Santos. A middleware framework for the web integration of sensor networks. In Gerard Parr et al, editor, *S-Cube 2010, Revised Selected Papers*, number 57 in LN ICSTE. Springer-Verlag, 2011.
16. Mirko Sonntag, Dimka Karastoyanova, and Ewa Deelman. Bridging the gap between business and scientific workflows. In *6thInt.Conf.e-Science*. IEEECS, 2010.
17. Petia Wohed, Wil M.P. van der Aalst, Marlon Dumas, Arthur H.M. ter Hofstede, and Nick Russell. Pattern-based analysis of bpmn, 2005. TR Queensland UT.