Execução de Fluxos de Trabalho com Simulação de Redes de Sensores

Duarte Vieira e Francisco Martins

Faculdade de Ciências da Universidade de Lisboa
LaSIGE & Departamento de Informática
Edifício C6 Piso 3, Campo Grande
1749 - 016 Lisboa, Portugal
dvieira@lasige.di.fc.ul.pt, fmartins@di.fc.ul.pt

Resumo As redes de sensores têm ganho relevância nas mais variadas áreas, com especial ênfase na monitorização ambiental e industrial e, mais recentemente, na logística. A informação recolhida do meio ambiente (pelas redes de sensores) pode influenciar o decurso dos fluxos de trabalhos destas áreas, pelo que, quando estes são representados em sistemas de gestão de fluxos de trabalho, testar o sistema como um todo pode tornar-se bastante complicado. Geralmente os testes efectuados nestes sistemas fazem uso de informação de registos de execuções de fluxos de trabalho anteriores. Alternativamente, os testes podem ser efectuados recorrendo a simulações de aplicações de redes de sensores. Para além de cobrir as situações descritas no caso anterior, esta abordagem permite testar novos fluxos, bem como testar variações introduzidas nos fluxos de trabalho por eventos do meio ambiente. Este artigo descreve uma forma de integrar plataformas já existentes com o objectivo de introduzir a simulação de redes de sensores nos testes de fluxos de trabalho.

1 Introdução

Uma rede de sensores sem fios é composta por uma colecção de dispositivos capazes de medir um determinado campo escalar ou vectorial e cuja comunicação entre os vários nós é feita sem fios. Numa rede de sensores podem existir nós com a capacidade de actuar sobre o ambiente (actuadores), bem como uma (ou mais) estação-base, responsável por processar os dados provenientes dos sensores e, quando necessário, (re)configurar o comportamento da rede.

As redes de sensores são um tópico que no passado recente cresceu em atenção quer por parte de empresas, quer por parte de grupos de investigação. Os desafios colocados ao nível do *hardware*, como a miniaturização dos dispositivos ou o aumento da capacidade da bateria e do alcance da comunicação sem fios, ombreiam com os desafios colocados ao nível do *software*, particularmente no que concerne a sistemas operativos e a linguagens de programação para estes dispositivos. Em ambas as áreas têm-se registado avanços importantes [3, 13, 20].

INForum 2010 - II Simpósio de Informática, Luís S. Barbosa, Miguel P. Correia (eds), 9-10 Setembro, 2010, pp. 599–610

As aplicações das redes de sensores são inúmeras e abrangem, por exemplo, desde a leitura de sinais vitais do nosso organismo (*body sensor networks*), passando por redes de monitorização de condições ambientais (*e.g.*, medição da qualidade do ar ou dos oceanos, detecção de fogos florestais), até a aplicações espaciais [2]. Uma das áreas de aplicação mais recentes das redes de sensores é a *Internet das Coisas e Serviços* (IoT), que consiste em integrar o *estado do mundo* visto pelos *olhos* de sensores em aplicações de mais alto nível, disponibilizadas na *Internet*. Desta forma, as aplicações podem beneficiar de observações realizadas no ambiente onde estão integradas e adequar o seu comportamento de acordo com os valores lidos. Por exemplo, uma aplicação de *domótica* pode estender o âmbito das suas funcionalidades se, para além de oferecer o tradicional agendamento de tarefas (*e.g.*, ligar ou desligar um dado dispositivo, ou subir ou descer persianas de acordo com um plano pré-estabelecido), reagir em função de condições ambientais ou de acordo com regras comportamentais dos habitantes da casa. Outra área de aplicação é a logística, onde, por exemplo, se pretende adequar um processo de entrega às condições da mercadoria que está a ser transportada e às condições de trânsito no trajecto até ao destino. Neste caso, as informações obtidas através dos sensores podem originar uma alteração no processo de entrega, podendo resultar na alteração da ordem de entrega das mercadorias.

Este tipo de aplicações que descrevem fluxos de trabalho são difíceis de testar, pois baseiam o seu comportamento em acontecimentos externos (do mundo) que são, no caso geral, não deterministas. A abordagem mais comum para testar estas aplicações consiste em repetir a execução de fluxos de trabalho guardados. Embora simples, esta abordagem padece de diversas enfermidades, a saber: (a) só permite testar fluxos de trabalho que não sofreram alterações relativamente ao traço que está guardado; (b) não permite testar novos fluxos de trabalho definidos; e (c) não permite testar novas variantes de fluxos actuais. Outra abordagem ao teste destas aplicações consiste em obter informação do ambiente através da simulação das redes de sensores, permitindo a criação de ambientes virtuais onde se pode estudar o comportamento dos sensores *per se*, da rede como um todo, e da aplicação. Esta abordagem tem como desvantagem a construção de um modelo para simular redes de sensores. À medida que o âmbito de aplicação das redes de sensores se alarga é necessário recorrer a simuladores, que por si só constituem um desafio, pois simular uma rede de sensores com comportamentos não triviais está longe de ser uma tarefa simples e rápida. Além disso, a integração dos resultados destas simulações com o sistema de gestão de fluxos de trabalho constitui outro problema a mitigar.

Em [19] descrevemos um processo que permite criar automaticamente um modelo de simulação de uma rede de sensores a partir de especificações de alto nível. No presente artigo iremos focar na interacção da simulação de redes de sensores com uma ferramenta de execução de fluxos de trabalho. Na secção seguinte apresentamos um cenário na área da logística que ilustra a utilização de sistemas de gestão de fluxos de trabalho integrados com redes de sensores

para avaliar as condições em que se processa uma entrega de matérias sensíveis à temperatura.

A organização do artigo é a seguinte: a Secção 2 aborda a simulação de redes de sensores, exemplificando com uma linguagem de programação e com um simulador; a Secção 3 apresenta alguns sistemas de gestão de fluxos de trabalho e elabora na integração da simulação de redes de sensores com a execução de fluxos de trabalho; finalmente, na Secção 4 concluímos o artigo e indicamos algumas direcções para trabalho futuro.

2 Simulação de Redes de Sensores no Contexto dos Fluxos de Trabalho

A simulação de redes de sensores permite testar não só aspectos como a transmissão de sinais, ou a medição de grandezas físicas, mas também aspectos de mais alto nível, como protocolos de comunicação e execução de aplicações. A simulação tem, portanto, grande interesse em qualquer área de actividade em que as redes de sensores sejam utilizadas.

Consideremos o seguinte cenário que descreve um fluxo de trabalho integrado com recolha de condições ambientais, obtida por sensores: um camião parte de Lisboa com dois contentores de vacinas, cada um equipado com um sensor capaz de medir a temperatura. Em ambos a temperatura não deve exceder os 10°C, sob pena de colocar em risco a qualidade das vacinas. O primeiro contentor a ser entregue tem como destino Coimbra e o segundo o Porto. O camião está equipado com uma estação-base que, periodicamente, pede a medição da temperatura aos sensores presentes no camião e que informa o centro de controlo, via GSM, caso alguma das leituras ultrapasse os 10°C. No centro de controlo um fluxo de trabalho é iniciado quando é recebida comunicação. Testar este fluxo de trabalho obriga a que sejam simuladas comunicações do camião. Todavia, se a simulação da rede de sensores estiver integrada com o fluxo de trabalho, tal não será necessário. Mais, a integração pode permitir a utilização de dados de simulação de redes de sensores com um grau de complexidade maior que o do exemplo anterior.

Em [19] propusemos um gerador de modelos de simulação de aplicações para redes de sensores, obtendo resultados encorajadores, tanto em tempo de simulação como em utilização de memória, para aplicações que correm em várias centenas de sensores. Nesta secção, apresentamos a linguagem para programar redes de sensores Callas e o gerador de modelos de simulação.

2.1 Callas

Callas [14] é uma linguagem de programação que tem por objectivo estabelecer uma base formal para o desenvolvimento de linguagens e sistemas de execução para redes de sensores. Pode ser utilizada por si só para programar redes de sensores ou como linguagem intermédia sobre a qual se possam compilar outras linguagens, com maior grau de abstracção.

A linguagem Callas é *type-safe*. Esta propriedade garante que programas bem tipificados não produzem erros em tempo de execução, algo de extrema importância no contexto das redes de sensores, em que os testes e a depuração são difíceis ou mesmo impossíveis de se fazer após a sua instalação num ambiente real.

Uma aplicação Callas é composta por interfaces (ficheiros `.caltype`), um programa por tipo de sensor (ficheiros `.callas`) e uma descrição da rede (ficheiro `.calnet`). Apresentamos agora uma aplicação Callas para a rede descrita acima. A interface types.caltype define os dois tipos de módulos usados nesta rede: Nil, o módulo vazio, e AlertTemp, um módulo com as funções sample, que não tem parâmetros e que retorna Nil, e alert, com os parâmetros mac (endereço MAC, *string*), time (tempo, *long*) e temp (temperatura, *double*) e que também retorna Nil.

```
# file: types.caltype

defmodule Nil: pass

defmodule AlertTemp:
  Nil sample()
  Nil alert(string mac, long time, double temp)
```

O ficheiro iface.caltype define a interface da rede. Todos os nós na rede serão do tipo Sensor, que estende o tipo AlertTemp e acrescenta a função listen.

```
# file: iface.caltype
from types import *

defmodule Sensor(AlertTemp):
  Nil listen()
```

Embora toda a rede implemente a mesma interface (garantia verificada em tempo de compilação), os sensores podem ter comportamentos diferentes, consoante a implementação da interface. O ficheiro node.callas contém o programa para os nós da rede, ou seja, para os sensores dos contentores. O programa importa as interfaces de iface.caltype, define o módulo m, instala-o (**store** m) e escalona a execução periódica da função listen. O módulo m implementa (a) a função listen, que consiste somente no comando **receive**, que lê mensagens da fila de entrada, (b) a função sample, que envia para a rede a chamada a alert(mac, time, temp), com os valores lidos e (c) a função alert, vazia.

```
# file: node.callas
from iface import *

module m of Sensor:
  def listen(self):
    receive
```

```
  def sample(self):
    mac = extern macAddr()
    time = extern getTime()
    temp = extern getTemperature()
    send alert(mac, time, temp)

  def alert(self, mac, time, temp):
    pass

store m
listen() every 30000 expire 36000000
```

O programa em sink.callas implementa a mesma interface, mas com um comportamento diferente. Na estação-base a função sample apenas envia para a rede a chamada a sample(). A função alert regista os valores recebidos e, caso a temperatura seja superior a 10°C, transmite-os por GSM. Para além do escalonamento periódico à função listen, faz um outro, à função sample, que por sua vez envia para a rede o chamada a sample().

```
# file: sink.callas
from iface import *

module m of Sensor:
  def listen(self):
    receive

  def sample(self):
    send sample()

  def alert(self, mac, time, temp):
    extern logString(mac)
    extern logLong(time)
    extern logDouble(temp)
    extern logString(" ")
    if temp > 10.0:
      extern sendGSM(mac, time, temp)

store m
listen() every 30000 expire 36000000
sample() every 60000 expire 36000000
```

2.2 Simulação de Aplicações Callas

Podemos categorizar os simuladores de redes de sensores como *específicos*, caso em que o simulador modela apenas uma arquitectura/sensor, ou como *genéricos*,

caso em que o simulador permite a modelação dos próprios sensores. O VisualSense [5] é um simulador genérico e de código fonte aberto baseado na plataforma de modelação e simulação Ptolemy II [9], desenvolvida pela UC Berkeley. Permite (a) simular todos os aspectos das redes de sensores referidos anteriormente, (b) simular redes em que os nós podem correr código diferente entre si, uma característica invulgar nos simuladores em geral [8] e (c) modelar e simular em modo gráfico. No Ptolemy II a modelação é feita utilizando componentes (chamados *actores*, seguindo o Modelo de Computação por Actores [1]) que interagem apenas por troca de mensagens.

O gerador de modelos apresentado em [19] permite a parametrização do número e disposição dos nós da rede de sensores, da aplicação que corre e dos modelos dos sensores e de rede. A simulação dos modelos gerados obteve bons resultados em termos de desempenho e escalabilidade, conforme se pode verificar na Figura 1.

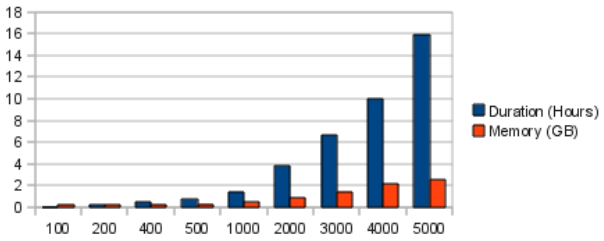

Figura 1. Duração da simulação e utilização da memória (abcissas) dado o número de sensores na rede (ordenadas).

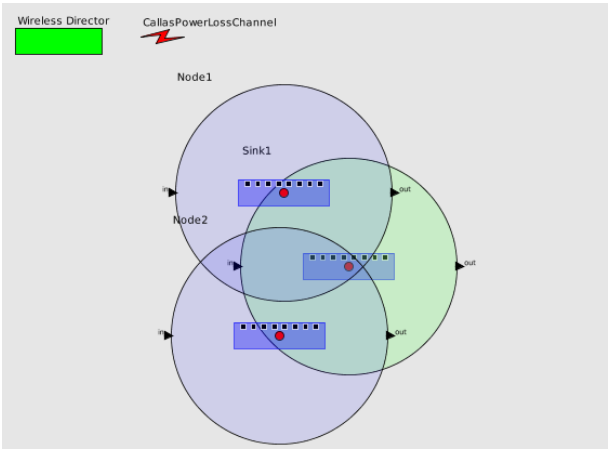

Figura 2. Rede de sensores no VisualSense.

O modelo de simulação para a aplicação definida na Secção 2.1 é gerado a partir do ficheiro network.calnet que, em adição à informação necessária à compilação do programa (interface a ser usada e o código de cada tipo de sensor), especifica, por exemplo, o número de nós e respectivas posições, o modelo do sensor (o formato persistente do Ptolemy II tem a extensão .moml) e o raio de comunicação. Com esta informação o gerador produz um modelo da rede que pode ainda ser editado no VisualSense, como se ilustra na Figura 2.

```
# file: network.calnet
interface = iface.caltype

sensor:
  code = node.callas
  size = 2
  range = 50
  position = random 0, 0 to 10, 10
  template = containerSensor.moml

sensor:
  code= sink.callas
  size = 1
  range = 50
  position = explicit 0, 0
  template = gsmSink.moml

template = containerNetwork.model # modelo do nível de rede
```

3 Execução de Fluxos de Trabalho em Kepler com Integração de Simulação de Redes de Sensores em VisualSense

Os sistemas de gestão de fluxos de trabalho, como o Sistema YAWL [18], são habitualmente utilizados para analisar processos de negócio, sendo também utilizados na validação dos processos de negócio em tempo de desenho [15].

Os sistemas de gestão de fluxos de trabalho científicos são uma especialização do tipo mais geral que permite executar fluxos de trabalho científicos (por exemplo, um conjunto estruturado de operações sobre dados provenientes de medições de grandezas físicas). Entre os sistemas deste tipo estão o Taverna [10], o Triana [16] e o Kepler [4].

O Kepler suporta modelação e execução em modo gráfico, composição de fluxos de trabalho, computação distribuída e acesso a repositórios de dados e serviços *web*. É um sistema baseado na plataforma Ptolemy II, tal como o VisualSense, mais propriamente, o Kepler e o VisualSense são especializações do Ptolemy II. A execução dos fluxos de trabalho é determinada pelo domínio de computação. Por exemplo, no domínio *Synchronous Dataflow* (SDF), a execução decorre de uma forma síncrona e numa sequência pré-calculada, enquanto que no

domínio *Process Networks* (PN) a execução é paralela, em que um ou mais componentes correm em simultâneo. Já o domínio *Discrete Event* (DE) é indicado para fluxos de trabalho com noção de tempo e eventos.

Uma vez que existe interoperabilidade de actores e directores (que implementam os domínios de computação) entre os dois sistemas, é possível integrar modelos de simulação de redes de sensores nos fluxos de trabalho definidos em Kepler de forma bastante directa. Deste modo, obtém-se uma forma de incluir informação proveniente da simulação de redes de sensores na simulação de processos de negócio. Adicionalmente, se tirarmos partido da linguagem Callas e do gerador de modelos de simulação, poderemos facilitar todo o trabalho relacionado com a simulação de rede de sensores.

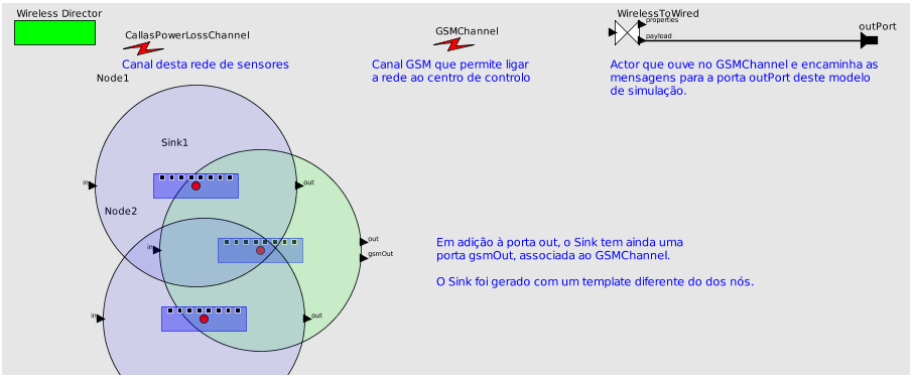

Figura 3. Modelo da rede de sensores para integração na execução do fluxo de trabalho

A Figura 3 apresenta o modelo de simulação da rede de sensores obtido a partir da aplicação da Secção 2. Na figura é possível identificar a azul os dois sensores de temperatura denominados por *Node1* e *Node2*. Estes sensores executam o código designado na Secção 2.1 pelo ficheiro node. callas , que faz com que enviem periodicamente o valor da temperatura de cada contentor para a estação-base, representada na figura a verde. Por seu lado, a estação-base, que executa o código do ficheiro sink . callas , notifica uma entidade central sempre que a temperatura em alguns dos contentores excede 10 graus centígrados. O modelo apresentado é gerado automaticamente em função da informação definida no ficheiro network. calnet . É possível observar o raio de alcance da comunicação dos sensores, bem como a sua posição relativa.

A comunicação entre os sensores e a estação-base ocorre através do canal CalasPowerLossChannel, que simula a comunicação sem fios entre os sensores no contentor. Este canal simula um meio de comunicação com perda de sinal e evita a não transmissão de mensagens repetidas. As portas in/out de todos os nós representados recebem/enviam mensagens utilizando o canal CalasPowerLossChannel. A comunicação da estação-base com um sistema central fora do camião é simu-

lada pelo canal GSMChannel. Este canal é o responsável pela ligação da simulação de sensores com a execução de fluxos de trabalho. A estação-base é a única que pode enviar mensagens no canal GSMChannel.

O fluxo de trabalho descrito pela Figura 4 calcula a melhor trajectória para as entregas, tendo em conta a temperatura do contentor e a localização do camião. A integração faz-se encapsulando o modelo da rede num actor do Kepler (TruckNetwork), que funciona como uma fonte de dados. A execução do fluxo de trabalho é, neste caso, despoletada por uma mensagem recebida do TruckNetwork. Num fluxo de trabalho diferente, a execução poderia decorrer iniciar-se por um outro evento, sendo alterada ao dar-se o evento proveniente da rede de sensores. No canto superior direito da Figura 3 encontra-se o actor WirelessToWired, que recebe as comunicações enviadas através do canal GSMChannel e as disponibiliza sob a forma de uma mensagem enviada para a porta de saída outPort do actor TruckNetwork representado na Figura 4. A mensagem da porta outPort do TruckNetwork é enviada para o actor MessageDisassembler, que dela extrai os valores de containerID e truckPosition, dados de entrada do fluxo de trabalho contido em CalculateBestPath, que calcula o melhor caminho a percorrer. Observe-se que o actor WirelessToWired (que liga a rede de sensores sem fios ao fluxo de trabalho) não está dependente da rede de sensores, nem do fluxo de trabalho; ele é somente um meio de transporte da mensagem.

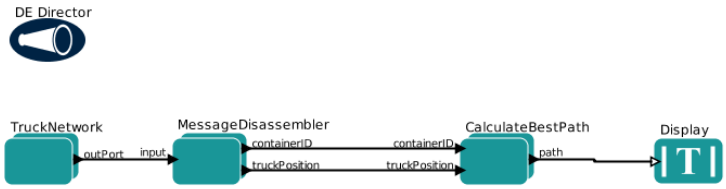

Figura 4. Fluxo de trabalho no Kepler. O actor TruckNetwork encapsula o modelo da rede da Figura 3. O actor CalculateBestPath encapsula o fluxo de trabalho de cálculo da trajectória.

As dificuldades da integração da simulação da rede de sensores no Kepler prendem-se com a (eventual) heterogeneidade dos domínios de computação. A simulação de aplicações Callas faz-se no domínio *Wireless*, uma extensão do *Discrete Event*, nem sempre adequado para a execução de fluxos de trabalho. Todavia, não deverão surgir dificuldades nos processos de negócio que estamos interessados em simular (empresariais), uma vez que são habitualmente orientados a eventos.

4 Conclusão e Trabalho Futuro

Neste artigo apresentamos uma forma de integrar a simulação de redes de sensores na execução de fluxos de trabalho, o que permite testar aplicações de mais alto nível baseadas em informação das *coisas*. A nossa proposta pretende integrar a simulação de redes de sensores da plataforma VisualSense [5] no sistema de gestão de fluxos de trabalho Kepler [4], tirando partido da interoperabilidade de componentes (actores) entre os dois sistemas, fruto de serem ambos extensões da plataforma Ptolemy II [9]. Para a criação de modelos de simulação de redes de sensores propriamente ditas, fazemos uso de um gerador (de modelos de simulação) que apresentámos anteriormente [19]. Pensamos que a nossa abordagem será uma mais valia na área de testes de aplicações (não só as baseadas de fluxos de trabalho) que dependam de valores recolhidos do ambiente porque permite validar fluxos de trabalho novos ou em que pretendemos experimentar novas variantes. Em contraste, o teste de fluxos de trabalho baseados em informação sobre execuções anteriores de fluxos de trabalho não se afigura tão flexível e completa como a que propomos.

Como trabalho futuro, identificamos desde logo a validação deste modelo de execução de fluxos de trabalho, bem como a obtenção de resultados que nos permitam avaliar a solução que propomos. Um ponto interessante que nos merece atenção no futuro é a interoperabilidade de fluxos de trabalho, isto é, a possibilidade de executar fluxos de trabalho, de uma forma distribuída, em dois ou mais sistemas de gestão de fluxos de trabalho distintos. A variedade de sistemas de gestão de fluxos de trabalho, motores e linguagens de descrição, dificulta a interoperabilidade dos fluxos de trabalho. Por exemplo, o Taverna [10] interpreta a linguagem Simple Conceptual Unified Flow Language [10] (SCUFL), o Kepler interpreta Modeling Markup Language [6] (MoML), o sistema YAWL [18] usa a linguagem Yet Another Workflow Language [17] (YAWL) e o Triana [16] interpreta, além do seu próprio formato, Business Process Execution Language [11] (BPEL). A acrescer à dificuldade decorrente da variedade de linguagens de descrição e de plataformas de execução, frequentemente as linguagens têm expressividades diferentes, pelo que a tradução entre elas nem sempre é possível, o que compromete a interoperabilidade por via da tradução de linguagens.

A interoperabilidade de fluxos de trabalho poderia ser conseguida através da padronização da linguagem de descrição/execução. Tem havido tentativas nesse sentido, por exemplo, o Workflow Management Coalition criou o XPDL [11] e a Microsoft e a IBM criaram o BPEL, ambos com o intuito de se tornarem padrão. Um outro caminho para a interoperabilidade de fluxos de trabalho é o da integração de motores de fluxos de trabalho de tal forma que seja possível correr cada fluxo no seu ambiente de execução, mas podendo interagir com outros, a correr noutros ambientes. Uma abordagem nesse sentido é apresentada por Kukla et al. [12]. Os autores vêem os sistemas de gestão de fluxos de trabalho como aplicações *legacy* embebidas num ambiente de Grid Computing, no caso, no GEMLCA [7] (Grid Execution Management for Legacy Code Applications).

Embora não tivéssemos abordado a disponibilização da informação de sensores via *web*, tal não se afigura complicado e vislumbramos que poderá ser feito facilmente com recurso a serviços *web*: teria de se criar uma interface que expu-

sesse o simulador VisualSense como serviço *web* de forma a poder ser acedido remotamente.

Agradecimentos

Os autores são parcialmente suportados pelo projecto CALLAS da Fundação para a Ciência e Tecnologia (PTDC/EIA/71462/2006).

Referências

1. G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.
2. I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.
3. I. F. Akyildiz, M. C. Vurany, B. Ozgur, and A. Weilian Su. Wireless Sensor Networks: A Survey Revisited . *Computer Networks*, 2005.
4. I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *SSDBM'04: Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 423–424. IEEE Computer Society, 2004.
5. P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao. Modelling of sensor nets in Ptolemy II. In *Proceedings of IPSN'04*, pages 359–368. ACM Press, 2004.
6. Christopher Brooks, Edward A. Lee, Xiaojun Liu, Stephen Neuendorffer, Yang Zhao, and Haiyang Zheng. Heterogeneous concurrent modeling and design in java (volume 1: Introduction to ptolemy ii). Technical Report UCB/EECS-2008-28, Electrical Engineering and Computer Sciences University of California at Berkeley, 2008.
7. T. Delaitre, A. Goyeneche, P. Kacsuk, T. Kiss, G. Terstyanszky, and S. Winter. Gemlca: Grid execution management for legacy code architecture design. In *EUROMICRO'04: Proceedings of the 30th EUROMICRO Conference*, pages 477–483. IEEE Computer Society, 2004.
8. E. Egea-Lopez, J. Vales-Alonso, A. Martinez-Sala, P. Pavon-Mariño, and J. Garcia-Haro. Simulation Scalability Issues in Wireless Sensor Networks. *IEEE Communications Magazine*, 44(7):64–73, 2006.
9. J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity - the ptolemy approach. *Proceedings of IEEE*, 91(2):127–144, Jan 2003.
10. D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(Web Server issue):729–732, 2006.
11. R. Ko, S. Lee, and E. Lee. Business process management (bpm) standards: A survey. *Business Process Management journal*, 15(5), 2009.
12. T. Kukla, T. Kiss, G. Terstyanszky, and P. Kacsuk. A general and scalable solution for heterogeneous workflow invocation and nesting. In *WORKS 2008: Proceedings of the Workflows in Support of Large-Scale Science, Third Workshop*, pages 1–8. Springer-Verlag, 2008.
13. L. Lopes, F. Martins, and J. Barros. *Middleware for Network Eccentric and Mobile Applications*, chapter 2, pages 25–41. Springer-Verlag, 2009.

14. F. Martins, L. Lopes, and J. Barros. Towards the safe programming of wireless sensor networks. In *Proceedings of ETAPS'09*, 2009.
15. A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, and C. J. Fidge. Workflow simulation for operational decision support. *Data Knowl. Eng.*, 68(9):834–850, 2009.
16. I. J. Taylor and B. F. Schutz. Triana - A Quicklook Data Analysis System for Gravitational Wave Detectors. In *Second Workshop on Gravitational Wave Data Analysis*, pages 229–237. Editions Frontières, 1998.
17. W. van der Aalst. Yawl: yet another workflow language. *Information Systems*, 30(4):245–275, June 2005.
18. W. M. P. van der Aalst, L. Aldred, M. Dumas, and A. H. M. Ter Hofstede. Design and implementation of the yawl system. In *CAiSE'2004*. Springer-Verlag, 2004.
19. D. Vieira and F. Martins. Automatic generation of WSN simulations: From Callas applications to VisualSense models. In *Proceedings of SENSORCOMM'2010*, 2010 (to appear).
20. E. Yoneki and J. Bacon. A survey of wireless sensor network technologies: Research trends and middleware's role. Technical Report UCAM-CL-TR646, University of Cambridge, 2005.