DSL e Metodologia para Construção de Aplicações Ubíquas

Taniro Rodrigues[1], Claudio Miceli[2], Flávia Delicato[2], Luci Pirmez[2], Paulo Pires[2], Priscilla Victor[1], Thais Batista[1]

[1]DIMAp - Universidade Federal do Rio Grande do Norte (UFRN)
Natal - RN - Brasil

[2]PPGI-iNCE/DCC/IM – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro - RJ -Brasil

{tanirocr, cmicelifarias, fdelicato, paulo.f.pires, pridnt thaisbatista}@gmail.com, luci@nce.ufrj.br

Abstract. Wireless Sensor Networks and Actuators (WSANs) are a major component of Ubiquitous systems. However, the complexity of programming such networks requires domain experts to know the specifics of each sensor platforms available, thus increasing the learning curve for developing applications for these networks. In this paper we report on the integration of two existing works that aim to facilitate WSAN application building and increase the effectiveness of the development process for such environments. This paper integrates: (i) a domain specific language, (ii) a method and infrastructure to develop ubiquitous applications. We evaluate the proposed integration through a comparative analysis and a proof of concept.

Resumo. Redes de Sensores e Atuadores sem Fio (RSASF) são um dos principais componentes de Sistemas de Computação Ubíqua. Porém, devido à complexidade de se programar para este tipo de redes é necessário que especialistas de domínio conheçam especificidades das plataformas de sensores disponíveis, aumentando a curva de aprendizado para o desenvolvimento de aplicações. Este trabalho integra trabalhos existentes voltados para facilitar a construção de aplicações para RSASF, as quais incluem: *(i)* uma linguagem específica de domínio para RSASF, *(ii)* uma metodologia e infraestrutura para desenvolvimento de aplicações ubíquas. A integração proposta foi avaliada através de uma análise comparativa e uma prova de conceito.

1. Introdução

As RSASFs são compostas por dispositivos de baixo custo, com fonte de energia e capacidades computacionais limitadas, dotados de capacidades de sensoriamento e comunicação sem fio. Atualmente desenvolvedores de aplicações que utilizam sensores ou atuadores precisam conhecer diversas características dos diferentes tipos de *hardware* e *software* que compõem a rede para construir aplicações a partir do uso de abstrações de baixo nível providas pelo sistema operacional do nó sensor. No entanto, o desenvolvimento deste é realizado por um *Especialista de Domínio (ED)*, responsável pelo conhecimento das regras de negócio presentes em um domínio alvo, e por isso deve ser idealmente transparente aos requisitos específicos de plataforma e rede. Por outro lado, um *Especialista de Redes (ER)* deve também atuar no desenvolvimento

destas aplicações promovendo a escolha de componentes que reflitam requisitos não-funcionais de importância fundamental para o funcionamento correto de qualquer aplicação nestas redes. Mais ainda, abordagens recentes sugerem o uso de redes de larga escala onde diferentes aplicações poderão ser executadas simultaneamente, dando origem ao conceito de RSASF compartilhadas (*Shared Sensor and Actuator Networks* [Farias et al. 2012]). A definição de aplicações neste contexto não poderá ser atrelada a uma plataforma específica, sendo necessária uma metodologia de desenvolvimento que abstraia características de plataforma e aumente a sinergia entre os diferentes tipos de desenvolvedores.

Além das dificuldades relacionadas ao conhecimento de domínio, plataforma e redes, RSASFs são fortemente afetadas por requisitos de *Qualidade de Serviço* (QoS), onde cada aplicação pode ter necessidades bastante distintas. Por exemplo, uma aplicação com requisito de *maximizar* o *tempo de vida da rede* para captar dados do ambiente ao longo de várias estações do ano precisa usar um protocolo de comunicação que faça uso mínimo do rádio. Por outro lado, aplicações que requerem respostas rápidas quando da detecção de um evento crítico (por exemplo, ocorrência de fogo) requerem protocolos que garantam atrasos mínimos, ainda que sob pena de aumentar o consumo de energia com transmissões rádio. Desta forma, fica claro que requisitos não-funcionais como tempo de vida, acurácia e atraso, dentre outros, possuem um papel fundamental no funcionamento destes tipos de sistemas e devem ser levados em conta durante a fase de projeto da aplicação. Portanto, ferramentas que forneçam meios de análise de requisitos não-funcionais em tempo de projeto da aplicação são potencialmente úteis nesse contexto. Tradicionalmente tal processo de análise pode ser realizado de duas maneiras: *(i)* a partir da implantação de aplicações nos nós para posterior execução da aplicação, a qual é uma tarefa muito custosa, pois em geral RSASF são compostas de dezenas a milhares de dispositivos; e *(ii)* a partir da execução de simulações que podem ser realizadas através de ambientes virtuais que sofrem de problemas incluindo: modelos simplistas, questões de escalabilidade, falta de customizações, dificuldade na obtenção de protocolos já existentes relevantes, etc. No entanto, em ambos os casos, não existe uma maneira de obter tal informação em *tempo de projeto*, pois tais abordagens representam uma barreira à aplicabilidade de análise de requisitos não-funcionais, algo que torna tais processos mais custosos e ineficientes.

Como solução para os problemas levantados, este artigo apresenta uma extensão de trabalhos anteriores, incluindo melhoraremos e a integração entre: *(i)* uma linguagem específica de domínio para a descrição de aplicações de RSASF [Dantas 2012] e *(ii)* uma metodologia e infraestrutura para desenvolvimento, análise de requisitos não-funcionais e geração automática de código-fonte para RSASF [Rodrigues et al. 2011]. Tal integração objetiva a substituição do Modelo Independente de Plataforma apresentado em [Rodrigues et al. 2011] por uma Linguagem Específica de Domínio (DSL, do inglês *Domain Specific Language*). Desta forma, *Especialistas de Domínio* poderão especificar suas aplicações através da seleção das características necessárias para que a aplicação atinja o objetivo esperado, sem se preocupar com as especificidades da rede ou da plataforma através do uso de uma DSL, denominada LWiSSy (*Domain **L**anguage for **Wi**reless **S**ensor and Actuators Networks **Sy**stems*). Os benefícios do uso desta linguagem serão avaliados através de um estudo comparativo realizado com outras DSLs existentes. A integração de LWiSSy com a abordagem *Model Driven Architecture* (MDA) [MDA 2013], denominada ArchWiSeN

(***Arch****itecture for* ***Wi****reless* ***Se****nsor and Actuator* ***N****etworks*), tornará possível a análise de requisitos não-funcionais durante a fase de projeto de uma aplicação para RSASF. Esta integração será avaliada através da utilização da linguagem, processo e infraestrutura MDA para o desenvolvimento de uma aplicação no domínio de *Smart Homes* [Soares et al. 2012] a fim de analisar o suporte que as soluções apresentadas oferecem para ganhos específicos a partir de boas decisões em tempo de projeto, facilitadas pelo uso da ArchWiSeN.

A Seção 2 deste artigo apresenta trabalhos relacionados. As Seções 3 e 4 apresentam a DSL e a abordagem MDA utilizadas. A Seção 5 compara a linguagem proposta com outros modelos da literatura e ilustra a integração entre a linguagem e o processo através de uma prova de conceito. A Seção 6 contém as considerações finais.

2. Trabalhos Relacionados

Diversos trabalhos atuais [Cetina et al. 2008; Fuentes and Gámez 2010; Losilla et al. 2007; Rodrigues et al. 2011; Shimizu et al. 2011] visam melhorar metodologias e ferramentas para desenvolvimento de aplicações para ambientes ubíquos. Porém, nestes trabalhos, as características de redes são especificadas junto às características do domínio, requerendo um maior nível de conhecimento dos especialistas e dificultando o reuso dos artefatos de software envolvidos. O uso de apenas uma visão, como é feito nas DSLs de [Losilla et al. 2007; Rodrigues et al. 2011] dificulta a modelagem das aplicações e não provê suporte ao *trade-off* entre o custo de prototipação e a capacidade de otimização em termos de desempenho. [Shimizu et al. 2011] propõe um o desenvolvimento de sistemas baseado em três modelos distintos de acordo com a granularidade da especificação (i) rede, (ii) grupo de nós e (iii) nó individual. Embora a construção de vários modelos seja proposta no trabalho, tais modelos descrevem apenas características estáticas das aplicações e não são capazes de descrever o comportamento das mesmas, o que é um fator crucial durante a modelagem de sistemas em RSASF. Além disso, os autores não consideram a participação de desenvolvedores com diferentes níveis de conhecimento e habilidades, e os meta-modelos não incluem várias características importantes de baixo nível, tais como o protocolo de comunicação, a topologia da rede e os tipos de dispositivos.

Em [Fuentes and Gámez 2010] é apresentado um Middleware customizado através de uma Linha de Produtos de Software para o desenvolvimento de aplicações para ambientes inteligentes. O *feature model* apresentado não separa características da rede de características do domínio. [Cetina et al. 2008] mostra uma abordagem onde a aplicação pervasiva poderia ser mantida e modificada de forma transparente ao usuário, porém as características de infraestrutura e organização da rede são não consideradas dentro os modelos apresentados. Em todas as abordagens levantadas, nenhuma se comprometeu de forma unilateral no processo de desenvolvimento, desde o projeto da aplicação até a geração de código, desta forma a integração da LWiSSy com ArchWiSeN cria uma abordagem única onde a linguagem e o processo se completam para formar uma abordagem que oferece aos desenvolvedores uma ferramenta completa para o desenvolvimento de aplicações para RSASF.

3. A Linguagem Específica de Domínio LWiSSy

A versão da LWiSSy apresentada neste artigo utiliza a UML [UML 2013] como meta-modelo base para que ocorra a integração com a ArchWiSeN. LWiSSy considera três

dimensões para o desenvolvimento de sistemas para RSASF: (i) *perfil do desenvolvedor* (especialista de redes ou domínio), (ii) *granularidade da especificação* (programação de rede, grupos de nós, ou nós individuais) e (iii) *projeto* (estrutural, comportamental e customização). LWiSSy usa a capacidade da UML de oferecer diferentes visões através de seus diagramas, tornando possível a separação da *dimensão de projeto* em três visões (i) *estrutural*, (ii) *comportamental* e de (iii) *customização* da aplicação, que serão oferecidas a cada desenvolvedor de acordo com a sua área de atuação, proporcionando maior abstração, alto grau de separação de interesses e reuso de artefatos de *software*.

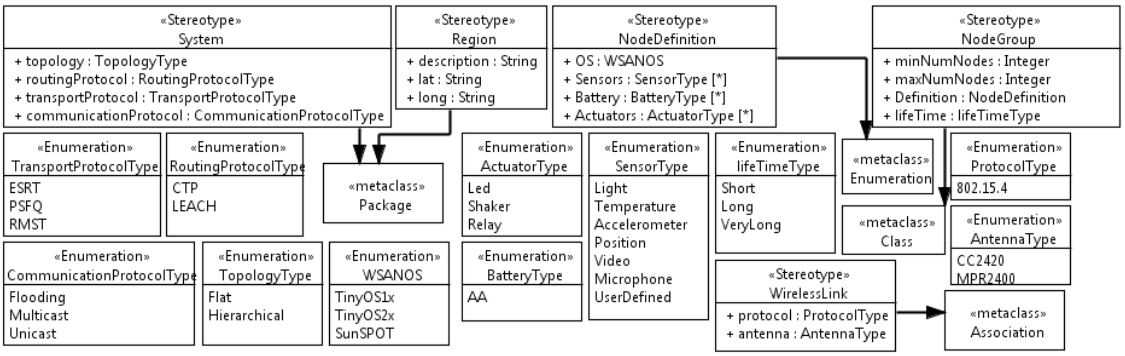

Figura 1. Estereótipos da visão estrutural.

Neste trabalho, foi construído um perfil UML para o contexto de aplicações RSASF, onde o conhecimento específico de domínio será representado através de modelos UML aprimorados com os estereótipos definidos neste perfil. Perfis são mecanismos de extensibilidade que permitem a modelagem de diferentes tipos de sistemas, domínios, métodos ou processos. As visões estrutural e comportamental da aplicação são modeladas pelo ED e representam o sistema em dois níveis de granulosidade diferentes: a especificação dos grupos de nós e a especificação de um nó individual que fazem parte da rede. A primeira visão trata da organização física dos grupos de nós (características como o número de regiões, localização, conexões entre os grupos etc.) que pertencem à aplicação ubíqua sendo modelada, ou seja, contemplam a arquitetura da aplicação (Figura 1). Na segunda visão são representados os elementos que expressam o comportamento de cada um dos componentes da aplicação (Figura 2). A visão de customização é modelada pelo ER. Sua função é adicionar informações sobre *hardware*, protocolos de redes e configurações utilizadas visando potencialmente melhorar seu desempenho.

LWiSSy utiliza para a especificação da visão estrutural um modelo UML de *Classes*. Esta visão define como o *Sistema*, estereótipo *<<System>>*, é especificado em termos de *Regiões*, estereótipo *<<Region>>*, definidas pela proximidade física e funcionalidade dos nós da rede (sensores, atuadores, sorvedouros), e recursos utilizados associados. Dentro das *Regiões* serão alocados os *Grupos de Nós*, estereótipo *<<NodeGroup>>*, que agrupam todos os nós dentro de uma mesma região que executam as mesmas aplicações. Diferentes *Grupos de Nós* podem ser conectados através de um *<<WirelessLink>>*.Para a especificação da visão comportamental, adotou-se um modelo UML de *Atividades*. O modelo é definido em termos de unidades funcionais, *<<FuncionalUnits>>*, que representam o comportamento da aplicação através de atividades comuns realizadas no domínio de RSASF, as quais podem ser comunicação, sensoriamento, fusão de dados, dentre outras. Por fim, as características da visão de customização podem ser incluídas por meio de estereótipos UML tanto nos

modelos da visão estrutural quanto nos modelos da visão comportamental como, por exemplo, topologia, protocolo de roteamento, sistema operacional, entre outros.

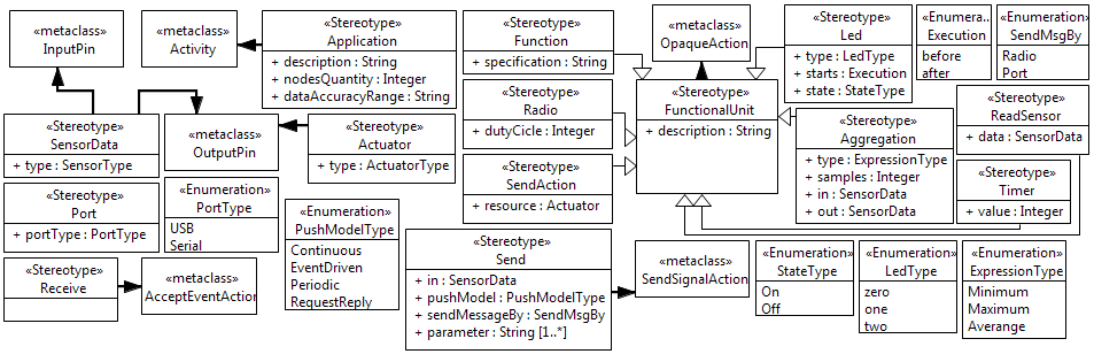

Figura 2. Estereótipos da visão comportamental.

4. Integração com a abordagem MDA ArchWiSeN

Esta Seção descreve a integração da abordagem MDA proposta em [Rodrigues et al. 2011] com a LWiSSy de forma a possibilitar uma infraestrutura e um processo de desenvolvimento de sistemas para RSASF a partir do uso dos artefatos (perfis UML - na forma de Modelo Independente de Plataforma ou **PIM**; Modelos Específicos de Plataforma ou **PSM**; transformações entre modelos ou **M2M**; e geração de código ou **M2T**) existentes na ArchWiSeN. A primeira atividade na Figura 3 é "1. Modelar com LWiSSy" é responsável pela modelagem da aplicação. Na segunda atividade "2. Escolher Plataforma", o ER avalia as plataformas de RSASF disponíveis e escolhe a que melhor atende aos requisitos da aplicação. Na atividade seguinte, "3. Aplicar Marcas de Customização", com o modelo da aplicação finalizado (tanto a visão comportamental quanto a visão estrutural), o ER pode aplicar marcas específicas de plataforma nos modelos, marcas que fornecem características da *visão de customização*.

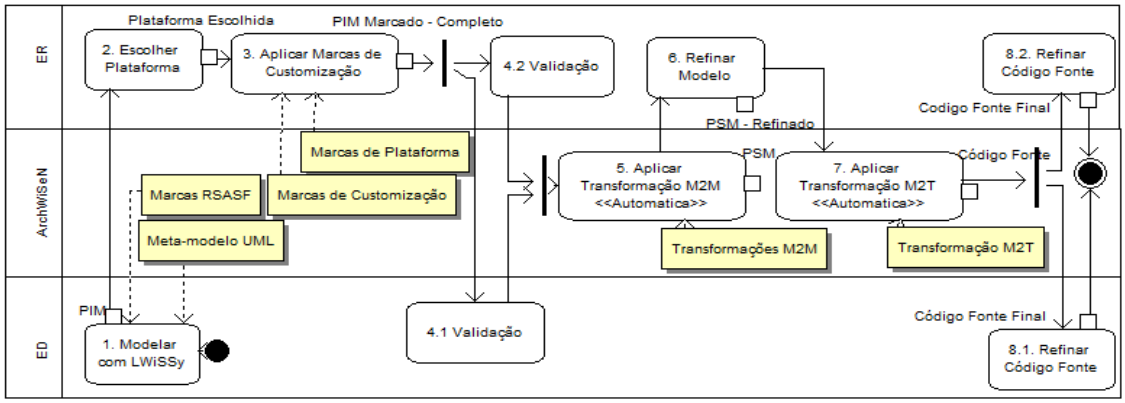

Figura 3. Diagrama de Atividades do processo de desenvolvimento de aplicações.

As atividades “4.1 e 4.2 Validação” usam as ferramentas disponíveis na ArchWiSeN para verificar os modelos desenvolvidos e analisar requisitos não-funcionais. A versão atual da ArchWiSeN oferece suporte para o análise do consumo de energia da aplicação através de cálculos realizados utilizando o modelo de energia da plataforma em questão e considerando as tarefas que estão sendo realizadas por cada um dos grupos de sensores. Com os dados oferecidos por esta ferramenta os desenvolvedores validam os modelos, para verificar se a aplicação modelada satisfaz os

requisitos. Na atividade "5. Aplicar Transformação M2M", realizada pela ArchWiSeN, um conjunto de transformações MDA é executado para mapear o modelo da aplicação do nível PIM (LWiSSy) para o nível PSM (plataforma). A atividade "6. Refinar Modelo" visa o refinamento manual do modelo PSM gerado automaticamente. Ela é realizada apenas pelo ER, já que envolve a manutenção de um PSM de implementação, ou seja, contém apenas características específicas de plataforma. Por fim, será possível gerar código através da atividade "7. Aplicar Transformação M2T". Existem ainda as atividades "8.1 e 8.2. Refinar Código Fonte", que são opcionais no processo. Nelas os desenvolvedores inserem manualmente funções específicas do seu domínio dentro do código, seguindo as regras sintáticas da linguagem da plataforma escolhida.

5.Avaliação

Esta Seção apresenta uma análise comparativa da LWiSSy com outras DSLs existentes (Subseção 5.1) e uma Prova de Conceito da integração proposta (Subseção 5.2). Em ambas as avaliações foram utilizadas RSASFs aplicadas a sistemas ubíquos, mais precisamente em *Smart Homes,* termo usado para definir uma casa ou edifício equipados com dispositivos inteligentes capazes de sensoriar e atuar sobre o ambiente, de modo a aumentar o conforto e a segurança dos usuários de modo não invasivo. Para a comparação, modelou-se uma aplicação para *Ambient Assisted Living (AAL*) [Delicato, F. et al. 2009] e para a prova de conceito foi utilizada uma aplicação AVAC (sistemas de Aquecimento, Ventilação e Ar Condicionado para controle de temperatura).

5.1. Análise Comparativa

Esta Subseção descreve uma análise comparativa entre LWiSSy e as DSLs propostas em [Losilla et al. 2007][Shimizu et al. 2011]. Na comparação, um sistema voltado para *AAL* é modelado utilizando as DSLs. Em particular, escolhemos um cenário baseado em um sistema para detecção de quedas acidentais para realizar a modelagem. A aplicação principal, chamada *AccidentalFallDetection System*, monitora um indivíduo e analisa informações coletadas por sensores, as quais podem ser interpretadas como uma queda dependendo da localização da pessoa, de sinais corporais, movimentos bruscos e o tempo em que a pessoa permanece em uma mesma posição. Se uma queda é detectada, ações como *enviar um alerta para o serviço de emergência* e *iniciar monitoramento de sinais vitais* são disparadas. Considerando tais requisitos, a RSASF para sistemas AAL pode ser composta de um nó dotado de acelerômetros instalado em crachá acoplado a pessoa monitorada, três sensores de imagens (câmeras), três nós sensores de posição fixos instalados nas paredes e atuadores com diferentes funções. Com o intuito de analisar os dados obtidos, faz-se necessária a definição de uma estação central responsável pelo processamento da informação recebida a fim de detectar a queda. O sistema de RSASF para o cenário descrito funciona da seguinte maneira. Os sensores instalados em uma parede formam a triangulação da posição do indivíduo utilizando sinais de RSSI (*Received Signal Strength Indication*) e podem detectar qualquer mudança na posição do mesmo. Enquanto isso, o sensor instalado no crachá detecta movimentos bruscos e determina se o indivíduo está de pé ou em movimento. Os dados capturados são então enviados para a central de controle. Se o indivíduo estiver efetuando um movimento de queda, a central envia um sinal para ligar as câmeras. A central recebe as imagens da sala e efetua o processamento com o objetivo de descobrir

se o indivíduo está apenas caminhando ou se está em queda. No caso de detecção de queda, a central entra em contato com um hospital ou serviço de emergência.

5.1.1. Modelagem do sistema utilizando diferentes DSLs

A primeira modelagem do sistema foi efetuada usando LWiSSy. No modelo estrutural definido pelo ED (Figura 4), observa-se que o sistema foi dividido em três regiões. *UserLocation Region* é formada pelos grupos de nós *BadgeGroup* e *PositionGroup*, responsáveis pela coleta de dados do ambiente e do indivíduo, respectivamente. Por sua vez, a região *UserVideoRegion* é composta pelo grupo de nós *VideoGroup*, que gerenciam as câmeras. Por fim, *CentralStationRegion* é composta por apenas um grupo de nós, *ProcessingGroup*, responsável pelo processamento dos dados coletados para detecção da queda e gerenciamento dos atuadores presentes na rede. Além disso, ainda foram definidos os tipos de recursos que serão utilizados em cada grupo de nós (*MovementSensor*, *PositionSensor*, etc.), a quantidade de nós presentes em cada um dos grupos de nós, as conexões existentes na rede e as aplicações que serão executadas.

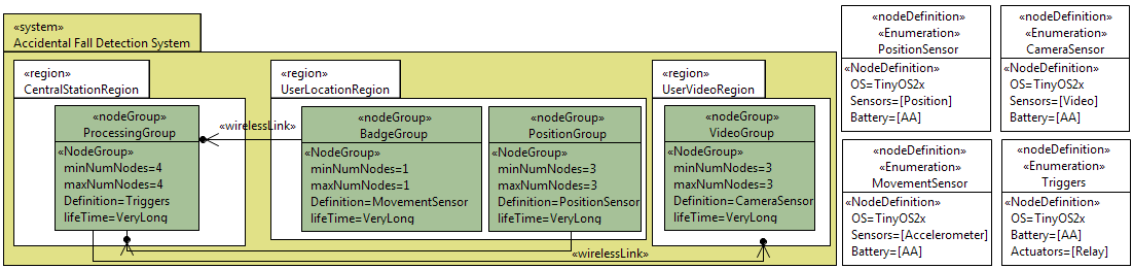

Figura 4. Modelagem estrutural do sistema *AccidentalFallDetection* com LWiSSy.

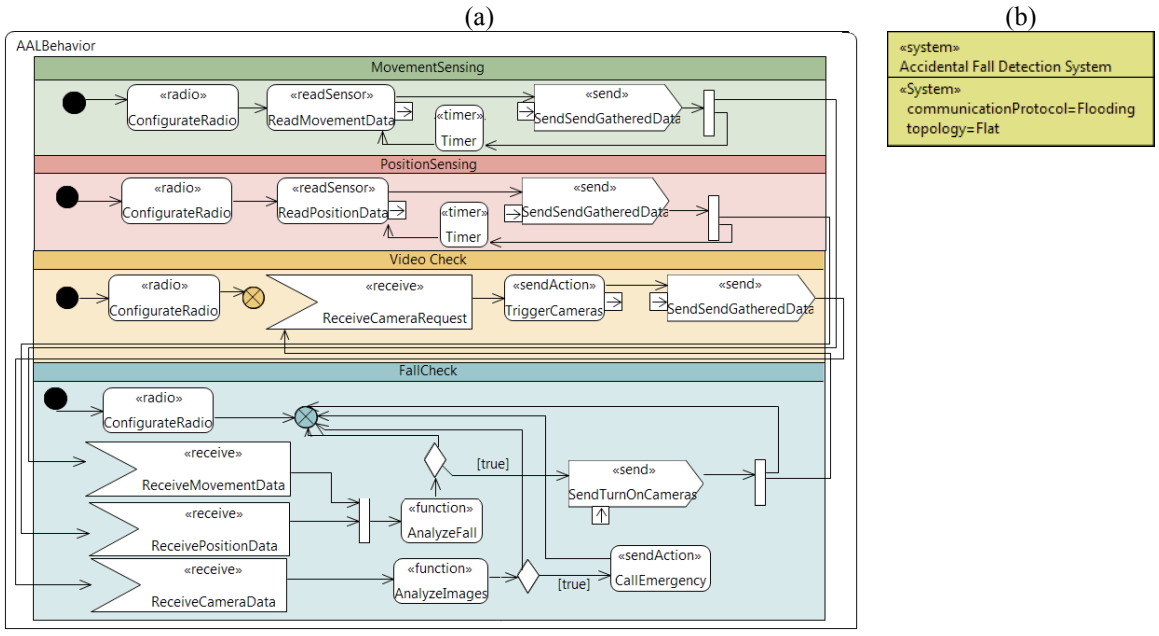

Figura 5. (a) Modelagem comportamental (4 aplicações) e (b) customização de *AccidentFallDetection* com LWiSSy.

No passo seguinte o ED realizou a modelagem comportamental das quatro aplicações que compõem o sistema, a saber, *MovementSensing*, *Position Sensing*, *VideoCheck* e *FallCheck* (Figura 5 (a)). Aplicação *MovementSensing* coleta dados sobre

a movimentação do indivíduo no ambiente e os envia à estação central. *PositionSensing* coleta dados sobre a posição do indivíduo no ambiente e também os envia à estação central. *VideoCheck*, iniciada pela estação central apenas se uma possibilidade de queda é detectada, liga as câmeras de vídeo e envia as imagens coletadas à estação central. Por fim, *FallCheck* recebe os dados sobre a movimentação e posição do indivíduo, efetua um processamento a fim de determinar se há possibilidade de queda e, caso haja, coleta dados das câmeras de vídeo para confirmar se ocorreu de fato uma queda. Em caso positivo, a aplicação aciona um hospital ou serviço de emergência. Por fim, o ER efetuou a modelagem de otimização do sistema, conforme mostrado na Figura 5 (b). Durante esta modelagem, foram definidos a topologia de rede e o protocolo de comunicação utilizado. Além disso, foram determinados quais os tipos de nós sensores que serão utilizados (nesse caso, todos os grupos utilizam nós da plataforma TinyOS [TOS 2013]).

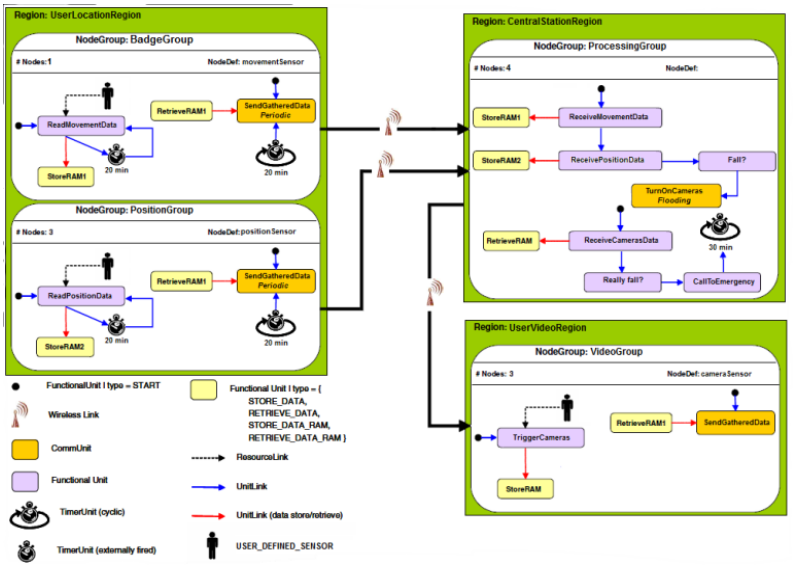

Figura 6. Modelo *AccidentalFallDetection* utilizando a DSL de [Losilla et al. 2007].

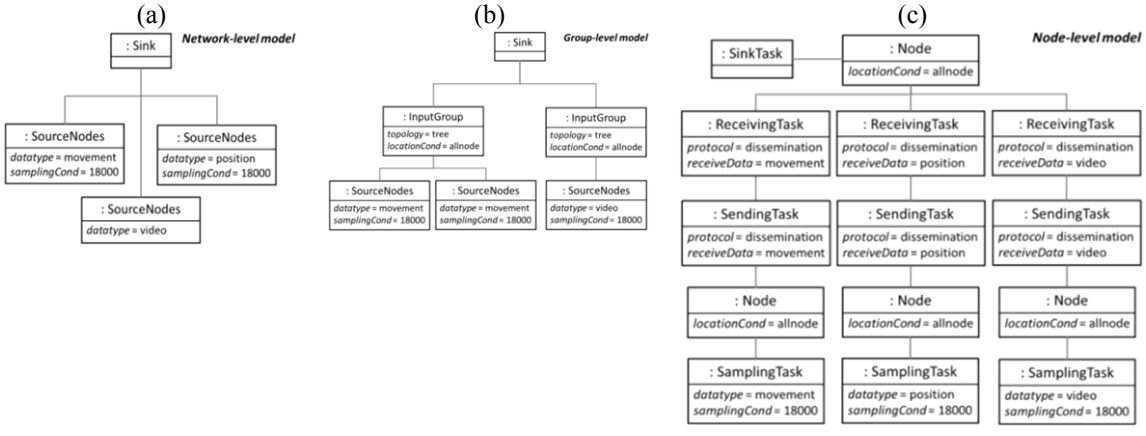

Figura 7. Modelo *AccidentalFallDetection* utilizando a DSL de [Shimizu et al. 2011].

A segunda DSL utilizada foi proposta por [Losilla et al. 2007], ilustrada na Figura 6. Durante esta modelagem foram definidas três regiões. *UserLocationRegion* é composta por dois grupos de nós: *BadgeGroup*, que efetua a coleta de dados de movimento do ambiente a cada vinte minutos e os envia à estação central, e

PositionGroup, que efetua a coleta de dados da posição do indivíduo no ambiente e os envia à estação central a cada vinte minutos. *UserVideoRegion* efetua a coleta das imagens capturadas pelas câmeras e as envia à estação central. A região *CentralStationRegion* recebe os dados coletados pelos grupos de nós *BadgeGroup* e *PositionGroup*, processa-os e avalia se há uma queda. Em caso positivo, as câmeras são ligadas e os dados são recuperados a fim de validar a queda e efetuar uma chamada para o serviço de emergência.

A última DSL na qual o sistema foi modelado com base no trabalho proposto por [Shimizu et al. 2011], conforme mostrado na Figura 7. Nesta modelagem foram utilizados três modelos que se distinguem pelo nível de granularidade considerado, podendo ser classificado como nível de rede, nível de grupos e nível de nós.

5.1.2. Análise

Após a modelagem do sistema *AccidentalFallDetection* com as três DSLs, é possível comparar o seu poder de expressividade de acordo com: (i) capacidade de otimização, (ii) separação de interesses, (iii) modelagem comportamental, e (iv) programação da rede em diferentes níveis de granularidade. Com relação à capacidade de otimização das DSLs em questão LWiSSy difere das demais por incluir atributos de otimização como última etapa no processo de desenvolvimento. Isto facilita o processo de modelagem do sistema uma vez que restringe a definição dos atributos de otimização apenas ao ER, provendo uma melhor separação de interesses. O segundo ponto a ser analisado também envolve a separação de interesses, ou seja, entre os diferentes perfis de desenvolvedor envolvidos na modelagem de sistemas de RSASF. Nota-se que apenas LWiSSy leva em consideração este fator. Na DSL proposta em [Losilla et al. 2007] há apenas um modelo englobando tanto características de domínio quanto de redes. Já em [Shimizu et al. 2011], é possível observar que características de domínio e de RSASF misturam-se ao longo dos modelos, causando uma dependência entre os especialistas envolvidos. Em LWiSSy, a divisão dos papéis dos especialistas é clara e não há conceitos de domínio e RSASF envolvidos em um mesmo modelo. A terceira característica analisada foi a capacidade de modelagem comportamental. Embora na DSL proposta por [Shimizu et al. 2011] seja possível definir quais atividades serão executadas pelos nós da rede, não há a descrição dos fluxos de caminho, não sendo possível definir qual será o comportamento correto de cada um dos nós. Em [Losilla et al. 2007] há a possibilidade de modelagem comportamental de cada um dos grupos de nós de maneira análoga á utilizada em LWiSSy. Todavia, a diferença entre essas DSLs consiste na separação entre conceitos comportamentais e estruturais que não foi considerada por [Losilla et al. 2007; Shimizu et al. 2011]. Por fim, a capacidade de programação da rede considerando diferentes níveis de granularidade (nível de rede, nível de grupo de nós e nível de nós) foi analisada. A DSL em [Losilla et al. 2007] difere das demais por não possibilitar uma clara separação entre os diversos níveis de granularidade, uma vez que apenas um modelo é utilizado ao longo do desenvolvimento de todo o sistema. LWiSSy e [Shimizu et al. 2011] utilizam três modelos que englobam diferentes níveis de granularidade.

5.2. Prova de Conceito

Para esta Prova de Conceito, o processo de desenvolvimento apresentado foi executado para a criação de uma aplicação para uma aplicação do tipo AVAC. A aplicação desenvolvida é composta de 4 tipos de nós diferentes: *(i) Nós Sensores Ordinários*

(NSO), que são os nós sensores dotados de unidades de sensoriamento, *(ii) Nós Atuadores* (NAT), que executam ações no ambiente monitorado, *(iii) Nós Sensores Decisores* (NSD), que recebem dados monitorados dos NSOs, tomam decisões sobre o ambiente monitorado, a partir de uma base de regras, e transmitem estas decisões para os NATs e (iv) *Nó Estação Base* (NEB) que serve para fins de manutenção e auditoria do sistema. A aplicação possui ainda um requisito não funcional que define que a rede deve ter um *tempo de vida útil longo*.

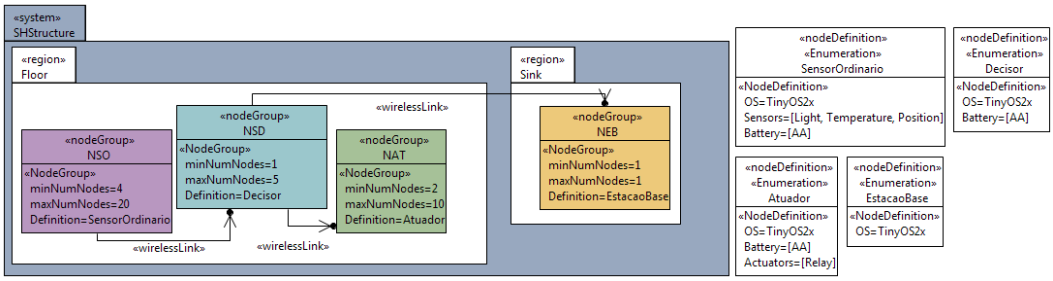

Figura 8. Visão estrutural da aplicação.

De acordo com o processo apresentado na Seção 4, o ED inicia a atividade "1. Modelar com LWiSSy" para desenvolver modelos UML que representem a aplicação sendo criada. A Figura 8 representa a visão estrutural da aplicação desenvolvida pelo ED, durante a atividade 1, e aperfeiçoada pelo ER. Serão necessárias duas regiões: *Floor*, contendo três grupos de nós (NSD, NSO, NAT), e *Sink* contendo um grupo de nó (NEB). Essa divisão entre *NodeGroups* permite que o ER posteriormente defina em detalhes o hardware que irá executar a aplicação em cada grupo, além do modo como os nós se comunicam entre grupos distintos. Como atividade paralela, o ER analisa os requisitos da aplicação-alvo e decide qual plataforma de sensor melhor os atende. Em seguida, o ER irá realizar a atividade "3. Aplicar Marcas de Customização", incluindo nos modelos fornecidos pelo ED as informações que refinam tais modelos visando incluir detalhes de plataforma e de funcionamento dos nós para a plataforma alvo (no caso, TinyOS). Após esta atividade, os desenvolvedores irão executar as transformações MDA e gerar código executável, seguindo as demais atividades (Seção 4).

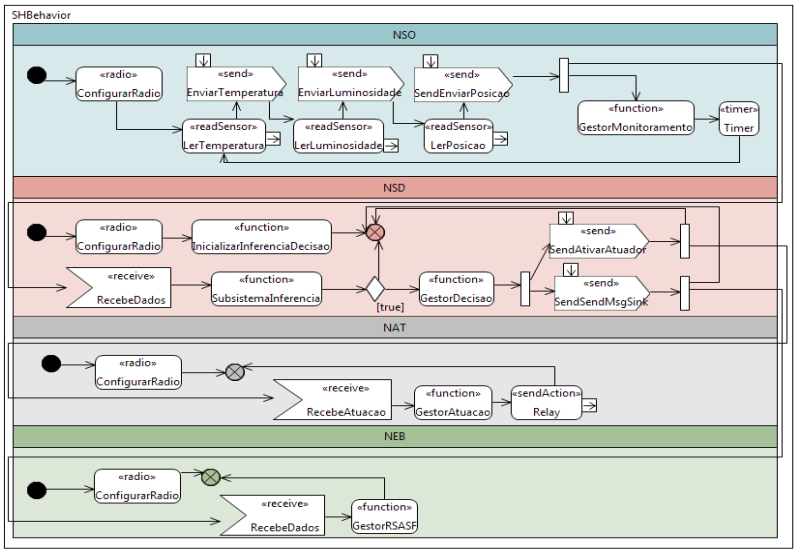

Figura 9. Visão comportamental da aplicação.

A Figura 9. ilustra as atividades, definidas pelo ED, a serem executadas pelos nós das categorias NSO, NSD, NAT e NEB. Na inicialização da aplicação, todos os nós irão executar a sua configuração de rádio. A seguir, o funcionamento de cada nó irá variar bastante de acordo com a funcionalidade de cada aplicação. Por questão de espaço nos limitaremos a falar da aplicação responsável pelo NSO. Nesta aplicação, após a configuração do rádio, os dados dos sensores são obtidos e enviados para o NSD. O processo será repetido de acordo com as configurações existentes no *GestorMonitoramento*, e ocorrerão de acordo com um timer pré-definido. Em um primeiro momento, o ED modela a aplicação realizando o envio dos dados obtidos pelos sensores a cada leitura conforme mostra a Figura 9, na partição NSO. Tal abordagem funciona de forma correta, porém após a fase de Validação (Figura 3), o ED percebe que a solução criada irá consumir muita energia, pois ocorrem muitos picos de corrente quando ocorre o uso do rádio, conforme mostra a Figura 10. Como solução para o problema apresentado, o ED modela novamente a aplicação, incluindo uma maneira de agregar os dados obtidos, e realizar apenas uma operação de envio de dados, conforme mostra a Figura 10. Desta forma, como ambas análises utilizam o mesmo *hardware* e o mesmo modelo de energia (MICAz nodes [Crossbrow 2013]), obtemos um cenário onde uma redução número de picos de corrente (que ocorrem durante o envio de dados) atua como um agente redutor de consumo de energia do nó, assim atingindo o requisito de duração da aplicação. Após esta etapa, o processo de desenvolvimento segue até a geração de código para a plataforma escolhida.

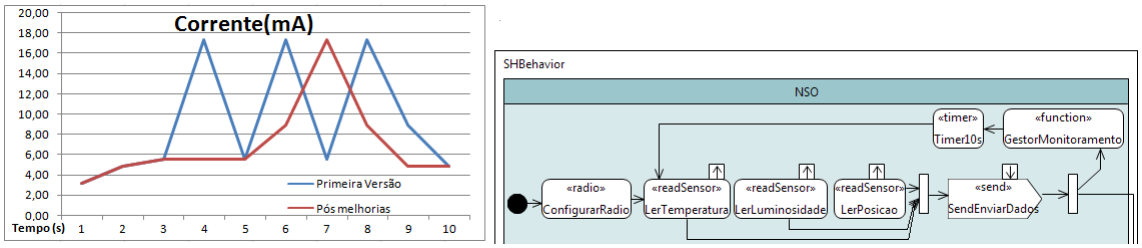

Figura 10. Solução para menor uso de energia.

5.2.1 Análise

Com a prova de conceito apresentada, foi possível demonstrar como o processo da ArchWiSeN integrado com a LWiSSy auxiliam os desenvolvedores durante a especificação de aplicações para sistemas de RSASF em geral e, mais especificamente, para sistemas Ubíquos. A atividade de validação dos requisitos do sistema é fundamental para que ocorra uma análise prévia do sistema ajudando os desenvolvedores a tomar melhores decisões durante a fase de projeto das aplicações e, desta forma, atingir os requisitos almejados.

6. Considerações Finais

A construção de aplicações fazendo uso da LWiSSy e ArchWiSeN provê todo um arcabouço para o desenvolvimento de aplicações para RSASF nos mais diferentes domínios. Tal abordagem possibilita ainda a divisão de responsabilidades entre os desenvolvedores. Outra importante contribuição é a capacidade de lidar com a alta heterogeneidade e as constantes mudanças tecnológicas na área de RSASFs, bem como de avaliar o comportamento da rede sob diferentes parâmetros e atendendo a requisitos não-funcionais em tempo de projeto.

Agradecimentos

Este trabalho foi parcialmente financiado pelas agências brasileiras: CNPq (bolsas 311363/2011-3, 470586/2011-7, 557.128/2009-9, 477223/2012-5, 473851/2012-1, 304941/2012-3, 485935/2011-2, DT 310661/2012-9), FAPERJ (E-26/102.961/2012 e E-26/170028/2008), FINEP (01.10.0064.00) e CAPES.

Referências

Cetina, C., Fons, J. and Pelechano, V. (sep 2008). Applying Software Product Lines to Build Autonomic Pervasive Systems. 2008 12th International Software Product Line Conference, p. 117–126.

Crossbrow (2013). MICAz Datasheet. http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf . Acesso em Abril, 2013.

Dantas, P. (2012). LWiSSy: uma linguagem específica de domínio para modelagem de sistemas de redes de sensores e atuadores sem fio. Dissertação de Mestrado. UFRN.

Delicato, F., Fuentes, L., Gámez, N. and Pires, P. F. (6 jun 2009). Variabilities of Wireless and Actuators Sensor Network Middleware for Ambient Assisted Living. International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, v. 5518, p. 851–858.

Farias, C., Pirmez, Luci, Delicato, Flavia Coimbra, Dos Santos, I. L. and Zomaya, A. Y. (oct 2012). Information fusion techniques applied to Shared Sensor and Actuator Networks. 37th Annual IEEE Conference on Local Computer Networks, p. 188–191.

Fuentes, L. and Gámez, N. (2010). Configuration Process of a Software Product Line for AmI Middleware. Journal of Universal Computer Science, v. 16, p. 1592–1611.

Losilla, F., Vicente-Chicote, C., Álvarez, B., Iborra, A. and Sánchez, P. (24 sep 2007). Wireless sensor network application development: an architecture-centric MDE approach. p. 179–194.

MDA (2013). Model Driven Architecture. http://www.omg.org/mda/. Acesso em Abril, 2013.

Rodrigues, T., Dantas, P., Delicato, Fl´via C, et al. (2011). Model-Driven Development of Wireless Sensor Network Applications. 2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing, p. 11–18.

Shimizu, R., Fukazawa, Y. and Honiden, S. (2011). Model Driven Development for Rapid Prototyping and Optimization of Wireless Sensor Network Applications Categories and Subject Descriptors. Development, n. ii, p. 31–36.

Soares, H., Pirmez, L., Delicato, F. and Farias, C. (2012). CONDE: Um Sistema de Controle e Decisão para Edifícios Inteligentes usando Redes de Sensores e Atuadores Sem Fio. SBRC,

TOS (2013). TinyOS. http://www.tinyos.net/. Acesso em Abril, 2013.

UML (2013). Unified Modeling Language. http://www.uml.org/ . Acesso em Abril, 2013.