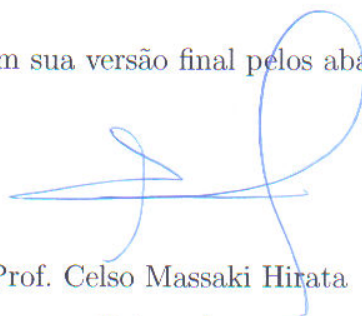


Tese apresentada à Pró-Reitoria de Pós-Graduação e Pesquisa do Instituto Tecnológico de Aeronáutica, como parte dos requisitos para obtenção do título de Mestre em Ciências no Programa de Pós-Graduação em Engenharia Eletrônica e Computação, Área de Informática

**Luciana Brasil Rebelo dos Santos**

**UM MODELO DE DADOS PARA  
CONFIGURAÇÃO DA SIMULAÇÃO NO  
DESENVOLVIMENTO DE APLICAÇÕES DE  
REDES DE SENSORES SEM FIO**

Tese aprovada em sua versão final pelos abaixo assinados:



Prof. Celso Massaki Hirata

Orientador

Prof. Vakulathil Abdurahiman (in memorian)

Orientador

Prof. Celso Massaki Hirata

Pró-Reitor de Pós-Graduação e Pesquisa

Campo Montenegro

São José dos Campos, SP - Brasil

2009

## Dados Internacionais de Catalogação-na-Publicação (CIP)

### Divisão de Informação e Documentação

Santos, Luciana Brasil Rebelo dos

Um Modelo de Dados para Configuração da Simulação no Desenvolvimento de Aplicações de Redes de Sensores Sem Fio / Luciana Brasil Rebelo dos Santos.

São José dos Campos, 2009.

93f.

Tese de Mestrado – Curso de Engenharia Eletrônica e Computação. Área de Informática – Instituto Tecnológico de Aeronáutica, 2009. Orientadores: Prof. Celso Massaki Hirata e Prof. Vakulathil Abdurahiman (in memorian).

1. Simulação de eventos discretos. 2. Sensores. 3. Comunicação sem fio. 4. Redes de comunicação. 5. Modelos de dados. 6. Engenharia eletrônica. 7. Computação. I. Comando-Geral de Tecnologia Aeroespacial. Instituto Tecnológico de Aeronáutica. Divisão de Ciência da Computação. II. Título.

## REFERÊNCIA BIBLIOGRÁFICA

SANTOS, Luciana Brasil Rebelo dos. **Um Modelo de Dados para Configuração da Simulação no Desenvolvimento de Aplicações de Redes de Sensores Sem Fio**. 2009. 93f. Tese de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

## CESSÃO DE DIREITOS

NOME DO AUTOR: Luciana Brasil Rebelo dos Santos

TÍTULO DO TRABALHO: Um Modelo de Dados para Configuração da Simulação no Desenvolvimento de Aplicações de Redes de Sensores Sem Fio.

TIPO DO TRABALHO/ANO: Tese / 2009

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias desta tese e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta tese pode ser reproduzida sem a autorização do autor.

---

Luciana Brasil Rebelo dos Santos

Praça Marechal Eduardo Gomes, 50 - Vila das Acácias

CEP 12.228-900 – São José dos Campos - SP - Brasil

# UM MODELO DE DADOS PARA CONFIGURAÇÃO DA SIMULAÇÃO NO DESENVOLVIMENTO DE APLICAÇÕES DE REDES DE SENSORES SEM FIO

**Luciana Brasil Rebelo dos Santos**

Composição da Banca Examinadora:

Prof. Carlos Henrique Quartucci Forster	Presidente	-	ITA
Prof. Celso Massaki Hirata	Orientador	-	ITA
Prof. Vakulathil Abdurahiman (in memorian)	Orientador	-	ITA
Prof. Roberto d'Amore	Membro	-	ITA
Prof. Nandamudi Lankalapalli Vijaykumar	Membro Externo	-	INPE

Aos homens da minha vida: Pedro Luiz, Oiram e Raymundo.

# Agradecimentos

Ao meu orientador, Prof. Celso Massaki Hirata, pela compreensão, confiança, seriedade e amizade, durante a realização deste trabalho.

À minha família, pelo carinho e incentivo. Agradecimento especial à minha mãe, por sua luta e ensinamentos de vida. Aos meus irmãos, pelo incentivo e apoio financeiro durante minha graduação.

Agradecimento especial ao meu colega Gian Ricardo Berkenbrock, pelas sugestões, troca de experiências, ajuda, incentivo e amizade. Obrigada!

Aos colegas de pós-graduação, Adalberto Dias, Carla Diacui M. Berkenbrock, Cinara G. Ghedini, Eduardo Barrios, Fábio F. Silveira, Frederico Alvares, Henrique G. Salvador, Luciano H.G. Marin, Milene Rigolin, Regina P.M. Dias, Thiago Trigo, Waldinez Araújo, pela amizade e momentos de descontração.

*“O homem é do tamanho do seu sonho.”*

— FERNANDO PESSOA

# Resumo

A simulação possibilita a realização de uma análise detalhada de Redes de Sensores Sem Fio (RSSFs) de forma rápida e econômica, tornando-se uma importante opção para estudo destas redes. Apesar do grande número de simuladores de RSSFs disponíveis, cada um deles foi projetado para atender necessidades específicas de seus criadores. Por causa dos aspectos singulares das RSSFs, os modelos existentes nos simuladores podem não ser precisos e completos para todas as áreas de simulação de RSSFs, como por exemplo, análise de escalabilidade, modelo de mobilidade, ambiente físico, consumo de energia e codições de tempo. Adicionalmente, a integração entre esses simuladores geralmente é pequena ou inexistente. Nesse contexto, este trabalho apresenta uma forma de modelagem abstrata de cenários de simulação de RSSFs, permitindo que um modelo de dados de configuração de simulação seja compartilhado por mais de um simulador, viabilizando a integração entre eles. A avaliação do modelo de dados proposto foi feita através do desenvolvimento de uma aplicação ilustrativa, onde o modelo de dados foi compartilhado por dois simuladores, que utilizaram um modelo de cenários e produziram adequadamente seus respectivos resultados de simulação.

# Abstract

*The simulation enables the realization of a detailed analysis of Wireless Sensor Networks (WSNs) in a fast and cheap manner, and for this reason it is an important option for studying such types of networks. Although a vast number of WSN simulators is available, each of them was designed to satisfy specific needs. Because of the unique aspects of WSNs, the existing simulator network models may not be accurate and complete, including all WSN simulation areas, such as scalability analysis, mobility model, physical environment, energy consumption, and weather conditions. Additionally, the integration among these simulators is generally weak or non-existent. In this context, this work presents an abstract modeling of WSNs simulation scenarios, allowing that different simulators share the same configuration data model, which enables integration among simulators. The evaluation of such data model was conducted through the development of an illustrated application, where the data model was shared by two simulators. The results of the experiments indicate that the simulators properly shared the data model and produced consistent results.*



# Sumário

LISTA DE FIGURAS . . . . .	12
LISTA DE TABELAS . . . . .	14
LISTA DE ABREVIATURAS E SIGLAS . . . . .	15
<b>1 INTRODUÇÃO . . . . .</b>	<b>16</b>
1.1 <b>Motivação . . . . .</b>	16
1.2 <b>Objetivos . . . . .</b>	18
1.3 <b>Contribuições . . . . .</b>	19
1.4 <b>Estrutura do Documento . . . . .</b>	19
<b>2 FUNDAMENTAÇÃO TEÓRICA . . . . .</b>	<b>21</b>
2.1 <b>Simulação Discreta . . . . .</b>	21
2.1.1 <b>Etapas de um estudo de simulação . . . . .</b>	22
2.2 <b>RSSFs . . . . .</b>	24
2.2.1 <b>Modelagem de Simulação para RSSF . . . . .</b>	26
2.2.2 <b>Objetivos de Simulação de RSSF . . . . .</b>	28
2.3 <b>Simuladores de RSSF . . . . .</b>	30
2.3.1 <b>TOSSIM . . . . .</b>	32
2.3.2 <b>COOJA . . . . .</b>	33

SUMÁRIO	10
2.3.3 OMNeT++	33
2.3.4 Castalia	34
2.3.5 NS-2	35
2.3.6 J-Sim	35
2.3.7 GloMoSim	36
2.3.8 ATEMU	36
2.3.9 AVrora	36
2.3.10 EmStar	37
2.3.11 OPNET	37
2.3.12 SENS	37
2.3.13 SENSE	38
2.4 Técnicas de Transformação de Modelos	38
2.5 Trabalhos Relacionados	42
2.6 Sumário	44
3 MODELO DE DADOS DE SIMULAÇÃO DE RSSF	46
3.1 Definição do MD	47
3.2 Descrição do MD	49
3.3 Utilização do MD	52
3.4 Transformação do MD em Modelo de Simulação	54
3.5 Sumário	55
4 EXEMPLO DE USO	57
4.1 Aplicação com Castalia	58
4.2 Aplicação com TOSSIM	60
4.3 Sumário	62

SUMÁRIO	11
<b>5 APLICAÇÃO</b>	64
5.1 Descrição	64
5.2 Desenvolvimento da Simulação	67
5.2.1 Requisitos da aplicação	67
5.2.2 Experimentos Realizados	69
5.3 Resultados Obtidos e Análise	72
5.4 Sumário	74
<b>6 CONCLUSÕES E COMENTÁRIOS FINAIS</b>	75
6.1 Conclusões	75
6.2 Limitações	76
6.3 Sugestões de Trabalhos Futuros	77
REFERÊNCIAS BIBLIOGRÁFICAS	78
ANEXO A – XML SCHEMA	84

# Lista de Figuras

FIGURA 2.1 – Versão simplificada do processo de desenvolvimento de simulação . . .	24
FIGURA 2.2 – Exemplo de uma RSSF implantada . . . . .	26
FIGURA 2.3 – Arquitetura de simulação de uma RSSF . . . . .	27
FIGURA 2.4 – Arquitetura do nó sensor . . . . .	28
FIGURA 2.5 – Visão Geral do Processo XSLT . . . . .	41
FIGURA 3.1 – Proposta do modelo de dados extensível . . . . .	48
FIGURA 3.2 – Ilustração do MD como árvore XML . . . . .	52
FIGURA 3.3 – Exemplo de descrição da topologia no MD . . . . .	52
FIGURA 3.4 – Utilização da XSLT para transformar arquivos XML em uma varie- dade de formatos . . . . .	55
FIGURA 4.1 – Regras para tratar a topologia no Castalia . . . . .	59
FIGURA 4.2 – Resultado após aplicar regras de transformação . . . . .	59
FIGURA 4.3 – Exemplos dos dois tipos de componetes TinyOS . . . . .	60
FIGURA 4.4 – Esqueleto do arquivo OSCILLOSCOPE.h . . . . .	61
FIGURA 4.5 – Parte da regra para tratar a topologia no TOSSIM . . . . .	62
FIGURA 4.6 – Resultado após aplicar regras de transformação . . . . .	62
FIGURA 5.1 – Resumo do Fluxo Produtivo da Cachaça . . . . .	65
FIGURA 5.2 – Disposição dos barris no galpão . . . . .	68

---

FIGURA 5.3 – Distribuição dos sensores . . . . .	70
FIGURA 5.4 – Tempo de vida da rede na simulação no Castalia . . . . .	72
FIGURA 5.5 – Tempo de vida da rede na simulação no TOSSIM . . . . .	73
FIGURA 5.6 – Tempo de vida da rede Castalia x TOSSIM . . . . .	74

# Lista de Tabelas

TABELA 2.1 – Modelos que as ferramentas permitem simular . . . . .	39
TABELA 3.1 – Parâmetros contidos no MD . . . . .	50
TABELA 5.1 – Parâmetros e seus valores para a definição do cenário de simulação .	71

# Lista de Abreviaturas e Siglas

CPU	<i>Central Processing Unit</i>
MAC	<i>Medium Access Control</i>
MD	<u>M</u> odelo de <u>D</u> ados para Configuração da Simulação
PC	<i>Personal Computer</i>
RSSF	Rede de Sensor Sem Fio
RX	Recepção de dados
SO	Sistema Operacional
TX	Transmissão de dados
XML	<i>eXtensible Markup Language</i>
XSLT	<i>XML Stylesheet Language Transformations</i>

# 1 Introdução

## 1.1 Motivação

Uma Rede de Sensor Sem Fio (RSSF) é desenvolvida para detectar fenômenos, coletar dados, processá-los e transmitir a informação coletada para usuários ([ILYAS; MAHGOUB, 2005](#)).

As RSSFs têm características peculiares, diferindo das redes convencionais em vários aspectos. Possuem restrições de energia, devem ter capacidade de auto-organização, normalmente têm um grande número de nós e há freqüente mudança na topologia devido a falhas nos nós, seja por problemas de comunicação ou término da bateria.

Para realizar o monitoramento, os nós sensores são colocados diretamente no ambiente. Nem sempre esses locais são de fácil acesso, como por exemplo, no topo de um vulcão ativo ([WERNER-ALLEN \*et al.\*, 2006](#)) ou nas águas do mar ([DUNKELS, 2007](#)). Nesses casos, depois que os sensores são instalados, não há mais como resgatá-los. Como consequência, a implantação da rede torna-se uma tarefa difícil, consumidora de tempo, e por isso, de alto custo.

Devido a essas características particulares, a simulação é essencial para o desenvolvimento de aplicações e teste de novos protocolos de RSSF, conforme mencionado por



Egea-Lopez *et al.* (2005). Muitos artigos publicados contêm resultados baseados apenas em simulação experimental (CURREN, 2005). As vantagens são: menor custo, pois existem vários simuladores de RSSFs de código aberto disponíveis, praticidade em testar o comportamento da rede utilizando diferentes parâmetros, possibilidade de realizar validação de código, testar redes de larga escala e, dependendo das restrições do projeto, a simulação pode até indicar se a implantação da rede é ou não viável.

Em virtude da crescente popularidade das RSSFs, existe um grande número de simuladores disponível, como por exemplo J-Sim (SOBEIH *et al.*, 2005), NS-2 (DOWNARD, 2004), OMNeT++ (MALLANDA *et al.*, 2005), Castalia (BOULIS, 2009), GloMoSim (BAJAJ *et al.*, 1999), TOSSF (PERRONE; NICOL, 2002), EmStar (ELSON *et al.*, 2003), COOJA (OSTERLIND *et al.*, 2006), TOSSIM (LEVIS *et al.*, 2003), Avrora (TITZER; LEE; PALSBERG, 2005), ATEMU (POLLEY *et al.*, 2004), Viptos (CHEONG; LEE; ZHAO, 2006), SENS (SUNDRESH; KIM; AGHA, 2004), SENSE (CHEN *et al.*, 2004) e OPNET (CHANG, 1999). Alguns grupos de simuladores possuem funções muito similares entre si, enquanto outros grupos possuem funções complementares.

Contudo, por causa de seus aspectos singulares e limitações, os modelos existentes nos simuladores de RSSFs podem não levar a uma demonstração completa de tudo que precisa ser verificado (PARK; SAVVIDES; SRIVASTAVA, 2001). Vários problemas são encontrados nos simuladores, como modelos super-simplificados para alguns objetivos de simulação, dificuldade para fazer customizações e também em obter protocolos relevantes já existentes (HANDZISKI *et al.*, 2003). Por causa da dificuldade em se obter um simulador que reúna todas as características pertinentes para um estudo completo de RSSFs, análises mais confiáveis e precisas podem ser obtidas combinando os modelos de diversos simuladores.

Ocorre que os simuladores de RSSFs, em geral, usam de um conjunto de especificações de modelagem distintas. Além do mais, quase não existe integração entre eles. Para cada simulador utilizado, uma nova modelagem é requerida, sendo necessário realizar o procedimento de verificação e validação do modelo que, segundo [Balci \(1994\)](#), é uma tarefa que consome bastante tempo e custo.

Essa discussão mostra a necessidade de uma forma unificada de modelagem de dados para o desenvolvimento, que permita modelar, simular e implantar aplicações de RSSFs. A definição de um modelo de dados para configuração de simulação de RSSF que possa ser utilizada por vários simuladores, é o primeiro passo para iniciar essa situação, promovendo uma modelagem unificada. Além disso, é uma solução para evitar o retrabalho de verificação e validação de modelos, facilitando uma análise ou um estudo mais integrado de RSSFs.

## 1.2 Objetivos

O objetivo do trabalho é apresentar um modelo de dados para a configuração de cenários de simulação no desenvolvimento de aplicações de RSSF, que permite uma modelagem unificada. Esse modelo de dados vai permitir o compartilhamento do modelo de simulação pelos simuladores de RSSF. O modelo foi obtido utilizando-se do seguinte procedimento:

1. São estabelecidos os pontos em comum na modelagem de RSSFs de acordo com o que foi observado nos simuladores estudados - os simuladores estudados são mostrados na [Seção 2.3](#). Este passo permite a definição do modelo de dados.
2. Aplicando regras de transformação no modelo comum de dados é possível extrair o modelo de simulação de simuladores distintos. Este passo permite a utilização do

modelo de dados.

### 1.3 Contribuições

Nos últimos anos, as RSSFs têm ganhado grande destaque e não é novidade que existe uma quantidade considerável de simuladores de RSSFs. Durante a elaboração do presente trabalho, na revisão de simuladores de RSSFs ativos, foram encontrados mais de trinta disponíveis. Para suprir seus próprios propósitos, muitas pesquisas resultaram no desenvolvimento de simuladores específicos. Cada simulador permite endereçar um subconjunto de objetivos de simulação. Como consequência, tornou-se comum a necessidade de se utilizar mais de um simulador para estudar uma aplicação de RSSF de forma mais completa. Contudo, é quase inexistente a integração entre os modelos dos simuladores.

O trabalho levanta a necessidade de uma maneira organizada no desenvolvimento de aplicações de RSSF, e propõe uma forma para iniciar esse processo, através de uma modelagem comum de dados dos simuladores, criando um modelo de dados para configuração de cenários de simulação de RSSF.

### 1.4 Estrutura do Documento

O restante deste documento está organizado da seguinte forma: no Capítulo 2 são discutidos alguns conceitos necessários ao entendimento do contexto deste trabalho, e que fundamentam teoricamente a solução proposta.

O Capítulo 3 apresenta o modelo de dados unificado para configuração de cenários de simulação no desenvolvimento de aplicações de RSSF. É feita uma descrição do modelo,

mostrando como ocorre sua utilização. É explicado como foram definidos os parâmetros que o modelo engloba, além da transformação do modelo geral para modelo específico.

O Capítulo 4 mostra um exemplo de definição do modelo de dados, implementado para os simuladores Castalia e TOSSIM.

O Capítulo 5 descreve a aplicação realizada para analisar e validar o modelo de dados definido, utilizando um cenário real, gerando os resultados obtidos e análises para essa aplicação.

Por fim, no Capítulo 6 são apresentadas as contribuições obtidas e as direções futuras desta pesquisa.

## 2 Fundamentação Teórica

Neste capítulo são apresentados os conceitos fundamentais que são a base para a pesquisa realizada. O capítulo está organizado da seguinte forma. A Seção 2.1 apresenta algumas definições importantes de Simulação Discreta. A Seção 2.2 discute sobre Redes de Sensores Sem Fio e sobre a simulação nessas redes. A Seção 2.3 mostra algumas plataformas para RSSF e descreve alguns dos principais simuladores de RSSFs. A Seção 2.4 apresenta algumas técnicas de transformação de modelos. A Seção 2.5 mostra os trabalhos relacionados e faz uma comparação com o presente trabalho. Por fim, a Seção 2.6 resume os principais pontos deste capítulo.

### 2.1 Simulação Discreta

De acordo com Banks (1998) simulação é a imitação da operação de um processo ou sistema do mundo real, por um determinado período.

A utilização de simulação tornou-se indispensável como solução de muitos problemas existentes. A simulação envolve a descrição total ou parcial de um sistema, e permite análise de seu comportamento, auxiliando no desenvolvimento de projetos de sistemas reais.

O *modelo de simulação discreta* pode ser definido como aquele onde o estado das variáveis modifica-se somente quando ocorre algum evento (BANKS, 1998). *Modelo* é a representação de um sistema. *Variáveis de estado* são a coleção de todas as informações necessárias para definir o que está acontecendo no sistema em um dado ponto no tempo. *Evento* é uma ocorrência que modifica o estado de um sistema.

O comportamento do sistema é estudado pela construção de um modelo de simulação. Para desenvolver um modelo, é preciso decidir quais elementos do sistema devem ser incluídos. Para tomar essa decisão, deve ser estabelecido um objetivo de modelagem do sistema, baseado em um problema apresentado.

Modelagem é usualmente apontada como o núcleo de qualquer estudo de simulação e envolve o uso de métodos computacionais e estatísticos para análise do que foi definido na estruturação do problema (PIDD, 2004).

### 2.1.1 Etapas de um estudo de simulação

Após analisar alguns conceitos de simulação, é preciso ter em mente que a modelagem é apenas parte de um esforço maior, que é o projeto e implementação de um estudo de simulação. De acordo com Banks (1998), Law e Kelton (2000), os passos que compõem um estudo de simulação são mostrados a seguir. É importante notar que a simulação é um processo iterativo pois eventualmente, é necessário voltar a passos anteriores.

1. *Formulação do problema*: cada projeto deve começar com a definição do problema a ser resolvido. É importante que a definição esteja clara.
2. *Determinação dos objetivos e planejamento global do projeto*: o objetivo indica as questões que devem ser respondidas pela simulação. O planejamento do projeto

deve incluir uma exposição dos vários cenários que serão investigados.

3. *Conceitualização do modelo*: o sistema do mundo real sob investigação é abstraído por um modelo conceitual. É recomendado que a modelagem se inicie de maneira simples e se torne mais complexa à medida que o modelo vai sendo desenvolvido.
4. *Coleta de dados*: a coleta de dados engloba especificação de requisitos, especificação de parâmetros de entrada, probabilidade de distribuição, e assim por diante. Há uma interação constante entre a construção de um modelo e a coleta dos dados de entrada necessários.
5. *Construção do programa de computador*: programar o modelo em uma linguagem de programação ou em um software de simulação.
6. *Verificação*: verificação diz respeito ao modelo operacional. Ela verifica se o programa de simulação está tendo o desempenho apropriado.
7. *Validação*: é a determinação de que o modelo conceitual é uma representação precisa do sistema real.
8. *Projeto dos experimentos*: para cada cenário a ser simulado, decisões precisam ser tomadas com relação ao número de execuções de simulação que devem ser efetuadas, a forma de inicialização, dentre outras.
9. *Execuções de produção e análise*: execuções de produção e suas análises subsequentes, são usadas para estimar medidas de desempenho dos cenários que estão sendo simulados.
10. *Documentação e relatório*: É fundamental que a simulação seja documentada de forma clara e concisa. Os resultados obtidos também devem ser documentados e

arquivados.

Balci (1994), Sargent (2004) propõem uma outra forma de demonstrar as etapas que envolvem um estudo de simulação, apresentando o ciclo de vida de simulação, mostrado de forma simplificada, na Figura 2.1. Essa proposta dá ênfase aos procedimentos de verificação e validação durante todo o ciclo de vida da simulação.

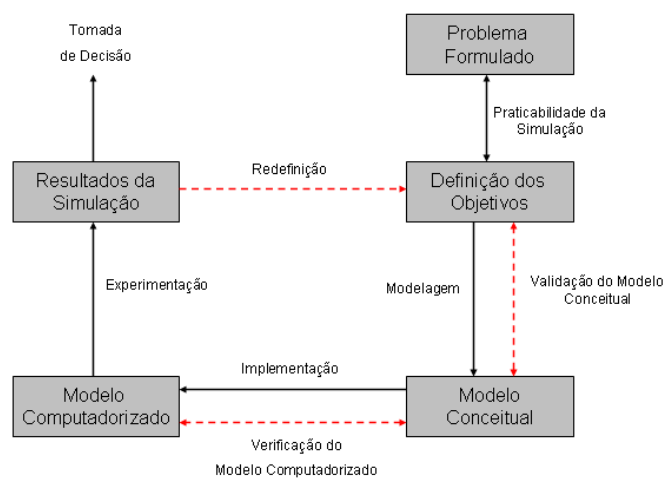


FIGURA 2.1 – Versão simplificada do processo de desenvolvimento de simulação

## 2.2 RSSFs

Nos últimos anos, avanços na área de micro-processadores, dispositivos de sensoria-mento e comunicação sem fio possibilitaram o desenvolvimento de redes de sensores sem fio – as RSSFs (AKYILDIZ *et al.*, 2002). Essas redes são providas de nós que podem sensoriar fenômenos físicos, executar pequeno processamento e realizar comunicação sem fio. Estes nós são chamados de nós sensores. Nós sensores são pequenos dispositivos com reduzida capacidade de processamento, comunicação e energia e de baixo custo.



As RSSFs podem ser vistas como um tipo especial de rede móvel *ad-hoc*. Em uma rede tradicional a comunicação é feita através de estações base de rádio. Em contrapartida, em uma rede móvel *ad-hoc* os elementos computacionais trocam dados diretamente entre si. Nas redes móveis *ad-hoc*, esses elementos computacionais podem executar tarefas distintas. As RSSFs tendem a executar uma função em conjunto onde os sensores provêm dados que são processados ou consumidos por nós especiais chamados de *sink* ou usuário (LOUREIRO *et al.*, 2003).

A principal característica das RSSFs é sua habilidade de monitorar o ambiente físico através do uso de vários nós sensores. Cada nó é equipado com uma variedade de sensores, tais como acústico, infravermelho, câmera, temperatura, sísmico, luminosidade, umidade e pressão. Os sensores são alimentados por baterias, geralmente não recarregáveis, o que torna a economia de energia essencial para o prolongamento do tempo de vida da rede.

Os nós são organizados de forma que pelo menos um dos sensores deve ser capaz de detectar um evento na região, processá-lo e tomar uma decisão se deve ou não comunicar o resultado para outros nós vizinhos. Estas atividades são controladas por algoritmos e protocolos, que também devem ter a capacidade de auto-organização, pois podem existir falhas nos nós que são causadas por danos físicos, término de energia ou influências do ambiente. Assim, a rede deve ter a capacidade de se reorganizar a cada falha para continuar sua operação e, ao mesmo tempo, tentar economizar energia. A comunicação entre os nós é efetuada através de antenas de curto alcance. A Figura 2.2 mostra como os elementos de uma RSSF normalmente são dispostos em um campo de sensoriamento.

Aplicações de RSSFs podem ser encontradas na agricultura; monitoramento de florestas, espécies e ecossistemas; controle de temperatura de ambientes; monitoramento da localização do inimigo em aplicações militares; presença de elementos químicos; dentre

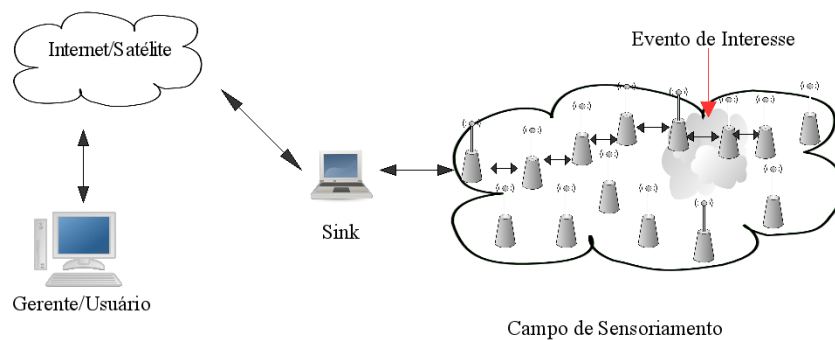


FIGURA 2.2 – Exemplo de uma RSSF implantada

outros exemplos.

Limitações de recursos como memória, processamento e bateria, a natureza dinâmica da implantação dos sensores no ambiente e dificuldade para realizar depuração, tornam o projeto e a implementação de protocolos e aplicações para estes dispositivos, uma tarefa desafiadora. Simulação pode ajudar a simplificar programação e testes de novos protocolos e aplicações para RSSFs, e por isso tornou-se uma técnica valiosa de verificação no desenvolvimento de sistemas para RSSFs, conforme mencionado por [Egea-Lopez et al. \(2005\)](#), [Levis et al. \(2003\)](#).

### 2.2.1 Modelagem de Simulação para RSSF

A identificação de elementos básicos de uma RSSF pode ajudar na definição de um campo comum para a integração de simuladores. Esta subseção descreve o modelo básico de simulação de uma RSSF que acredita-se ser comum para a maioria dos simuladores.

[Park, Savvides e Srivastava \(2001\)](#) propõem um ambiente para a simulação de RSSFs, incluindo novos componentes, que não estão presentes em simuladores de redes clássicas, como por exemplo, modelo detalhado de bateria, consumo de energia e ambiente. A maioria dos simuladores de RSSF utiliza arquitetura semelhante a essa. Esse ambiente

permite utilizar vários tipos de modelos para simular diferentes cenários. A arquitetura mostra o que um simulador deve prover para simular adequadamente uma RSSF.

A Figura 2.3, inspirada em Park, Savvides e Srivastava (2001), Egea-Lopez *et al.* (2005) mostra como uma RSSF é construída. São considerados os agentes, que são os geradores de eventos de interesse para os nós; o ambiente, que modela a propagação de eventos que são sensoriados pelos nós; os nós, que representam o equipamento físico de monitoramento; o canal sem fio que caracteriza a propagação de sinais de rádio entre os nós da rede; e o nó usuário ou nó *sink*, um nó especial que recebe dados da rede e os processa.

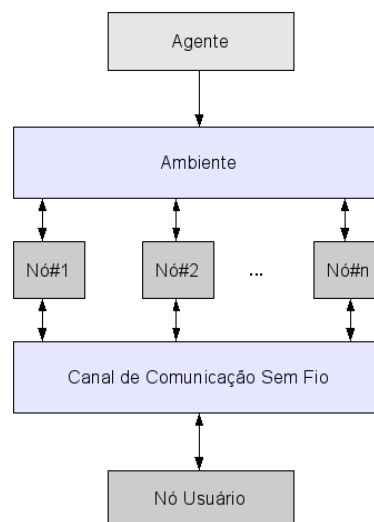


FIGURA 2.3 – Arquitetura de simulação de uma RSSF

O modelo de simulação do nó sensor é composto pela parte física do nó e por diversos elementos que influenciam seu comportamento. A parte física normalmente é composta pelos sensores físicos, modelo de energia (produção e consumo), equipamento de rádio e processador. Dependendo da aplicação, estes equipamentos podem ser alterados. Cada nó sensor, também, é equipado com a parte funcional, que compreende a pilha de protocolos de comunicação que interage com a camada de aplicação. A operação da pilha de

protocolos depende da interação com o modelo de energia. Por exemplo, o protocolo de roteamento deve considerar as restrições de bateria para decidir a rota dos pacotes. Além desses componentes, o nó sensor interage com o modelo de mobilidade, que controla sua posição; com o canal de sensoriamento, que representa o ambiente onde se propagam os eventos de interesse; e com o canal de comunicação sem fio, por onde os nós sensores se comunicam. A arquitetura do nó sensor pode ser visualizada na Figura 2.4.

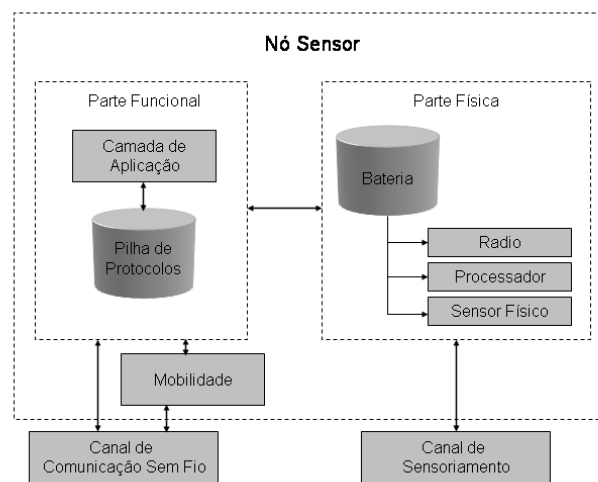


FIGURA 2.4 – Arquitetura do nó sensor

### 2.2.2 Objetivos de Simulação de RSSF

Muitos simuladores de RSSFs estão disponíveis para serem utilizados. Cada um dos simuladores possui um objetivo ou grupo de objetivos de simulação. As ferramentas de simulação de RSSF pode realizar simulações de:

- Meio físico: quando existem modelos para representar a propagação dos sinais em ambientes físicos específicos, como concreto, grama, água. Normalmente, os simuladores utilizam o ar como ambiente físico de propagação.
- Canal de comunicação: quando simuladores possuem modelos baseados em dados

coletados empiricamente ou com possibilidade de declarar valores em parâmetros que representam interferência, ruído, e assim por diante.

- Rádio: o modelo pode ser baseado em rádios reais de curto alcance ou através da inserção de valores nos parâmetros com diferentes estados (TX, RX, *idle*) e diferentes consumos de energia para cada estado.
- Sensoriamento: modelo de dispositivos de sensoriamento com parâmetros como interferência, consumo de energia, estados (*on*, *off*), e assim por diante.
- Bateria: modelos de tipos de baterias com parâmetros tais como valor inicial, taxa de decaimento, dentre outros.
- Emulação: quando o simulador provê suporte para conexão de sensores reais na simulação.
- Redes larga escala: quando é permitido simular um grande número de nós na rede de maneira consistente.
- Implantação de código: quando permite simular *deployment*.
- Sistema operacional: o código do sistema operacional pode ser simulado em um PC.
- Aplicação: permite simular aplicações, teste de novos protocolos, e assim por diante.
- Movimento dos nós: permite simulação de movimento de nós e fenômeno físico.
- Registros de CPU e memória: simulador de nível de instrução. Permite simular programas com o mesmo conjunto de instruções que é executado na plataforma.
- Fenômeno físico: modela as diversas formas de propagação de fenômeno, como um tanque em movimento, nuvem de poeira, temperatura ao longo do tempo, dentre

outros.

- Aplicações heterogêneas: permite simular aplicações diferentes na mesma simulação.
- Suporte gráfico: quando a ferramenta permite modelagem gráfica, provê suporte para animação, e assim por diante.

## 2.3 Simuladores de RSSF

Muitas pesquisas envolvendo redes de sensores resultaram no desenvolvimento de diversas ferramentas de simulação. Especialmente para RSSFs, em razão da diversidade de cenários de simulação, protocolos e elementos envolvidos, diferentes ferramentas de avaliação executam diferentes propósitos.

Por causa da evolução da tecnologia de micro sistemas eletrônicos, diversas plataformas computacionais são desenvolvidas. As principais plataformas de RSSFs suportadas pelos simuladores são:

- ***Mica Motes***: engloba os nós Mica ([HILL; CULLER, 2002](#)), Mica2 ([CROSSBOW-TECHNOLOGY, 2002](#)), Micaz ([CROSSBOW-TECHNOLOGY, 2004](#)) e TelosB ([POLASTRE; SZEWCZYK; CULLER, 2005](#)). A plataforma *Mica Motes* utiliza o sistema operacional TinyOS ([HILL \*et al.\*, 2000](#)), um SO baseado em eventos, cuja linguagem de programação é NesC ([GAY \*et al.\*, 2003](#)). Está se tornando a plataforma padrão de RSSF, segundo [Egea-Lopez \*et al.\* \(2005\)](#).
- ***Nymph*** ([ABRACH \*et al.\*, 2003](#)): é inspirado na arquitetura do Mica possuindo o mesmo processador e transceptor de rádio. Utiliza o sistema operacional MANTIS ([ABRACH \*et al.\*, 2003](#)).

- **EYES** ([HOESEL et al., 2003](#)): possui memória externa, bateria e placa de sensores similares aos do Mica. Utiliza o sistema operacional PeerOS ([MULDER, 2003](#)).
- **BEAN** ([VIEIRA, 2004b](#)): desenvolvido pela Universidade Federal de Minas Gerais. Utiliza o Yatos ([VIEIRA, 2004a](#)), um sistema operacional dirigido por eventos desenvolvido para o BEAN, que possui os conceitos de eventos e tarefas, similares ao do TinyOS.
- **TmoteSky** ([SKY, 2007](#)): utiliza o sistema operacional Contiki ([SWEDISH-INSTITUTE-COMPUTER-SCIENCE, 2005](#)), outro exemplo de SO, escrito na linguagem de programação C.
- **MSB** ([BAAR et al., 2006](#)): também utiliza o Contiki como sistema operacional. Provê sensores de temperatura e umidade relativa.

Os simuladores de RSSFs podem ser classificados de acordo com seu nível de operação: simuladores de aplicação, simuladores de sistema operacional e simuladores de nível de instrução. Simuladores de aplicação permitem verificar o desempenho de algoritmos e protocolos em uma primeira ordem de validação, antes de implementá-los em uma plataforma específica. Simuladores de sistema operacional executam o código que será executado no nó sensor, ou seja, o SO é emulado em um PC. Simuladores de nível de instrução executam programas com o mesmo conjunto de instruções que é executado em uma plataforma de sensoriamento. O simulador de nível de instrução simula a execução do código detalhadamente, onde são observados os registros da CPU e memória. Tais simuladores são utilizados quando a aquisição do hardware necessário é difícil ou inviável, como em projeto ou prototipação de microprocessadores e emulação de arquiteturas legadas ([TITZER; LEE; PALSBERG, 2005](#)).

Nesta seção são mostrados em maior detalhe os simuladores que foram investigados durante a realização deste trabalho. Estes são alguns dos simuladores mais importantes de RSSFs, por serem os mais citados na literatura de revisão de simulação de RSSF.

### 2.3.1 TOSSIM

TOSSIM (LEVIS *et al.*, 2003) (*TinyOS SIMulator*) é o simulador do sistema operacional TinyOS (HILL *et al.*, 2000), o SO de RSSF mais conhecido. Um programa TinyOS é um grafo de componentes, onde cada componente é uma entidade computacional independente. O componente TinyOS tem uma estrutura de variáveis privadas que podem ser referenciadas apenas por esse componente. Componentes têm três abstrações computacionais: comandos, eventos e tarefas. Comandos e eventos são mecanismos para comunicação inter-componentes, enquanto tarefas são utilizadas para expressar concorrência intra-componente.

TOSSIM tira vantagem da estrutura do TinyOS para gerar simulação de eventos discretos diretamente dos grafos de componentes TinyOS. Ele executa o mesmo código que é executado nos dispositivos de redes de sensores. Ao invés de compilar a aplicação do TinyOS para o *mote*, usuários podem compilá-la para o ambiente do TOSSIM, que é executado em um PC.

TOSSIM permite que usuários depurem, testem e analisem algoritmos em ambientes controlados, podendo simular milhares de nós com diferentes configurações de rede. TOSSIM possui algumas limitações, por exemplo, ele não captura consumo de energia; outra restrição é que todos os nós devem rodar o mesmo código, então não é possível avaliar alguns tipos de aplicações heterogêneas, ou seja, com códigos que têm assimetria textual.



TOSSIM foi escrito em C++ e suporta duas linguagens de programação para configuração da simulação: Python e C++.

### 2.3.2 COOJA

COOJA (ÖSTERLIND, 2006; ÖSTERLIND *et al.*, 2006) (*COntiki Os JAva*) é o simulador do sistema operacional Contiki (SWEDISH-INSTITUTE-COMPUTER-SCIENCE, 2005), que permite a simulação da aplicação e do sistema operacional. COOJA pode simular redes onde os nós sensores podem ser de diferentes plataformas ao mesmo tempo, como TmoteSky (SKY, 2007) e MSB (BAAR *et al.*, 2006). A linguagem utilizada em seu arquivo de configuração é XML.

### 2.3.3 OMNeT++

OMNeT++ (VARGA *et al.*, 2001) (*Objective Modular NETwork Test-bed in C++*) é um simulador de nível de aplicação. É voltado para simulação de redes em geral. Possui código aberto e utiliza a construção de blocos chamados módulos para implementar sua simulação. Os módulos podem ser simples ou compostos. Módulos simples são utilizados para definir algoritmos e são os mais baixos níveis da hierarquia. Módulos compostos são coleções de módulos simples os quais interagem uns com os outros através de troca de mensagens.

A topologia da rede de sensores é derivada do conceito de módulos simples e compostos. As camadas (por exemplo, camada de rede, MAC, física) de um nó se comportam como módulos simples e o nó sensor se comporta como um módulo composto e todos os nós sensores constituem a rede de sensores descrito como Módulo do Sistema.

OMNeT++ não é voltado para RSSFs, mas foram construídas extensões para simulação de RSSF, baseadas no OMNeT++, como por exemplo o Castalia (BOULIS, 2009) e MiXim (KöPKE *et al.*, 2008).

### 2.3.4 Castalia

Castalia (BOULIS, 2009) é um simulador de nível de aplicação especializado em RSSF baseado no OMNeT++ (VARGA *et al.*, 2001). As principais características do Castalia são:

- avançado modelo de canal de comunicação, baseado em dados coletados empiricamente. Modela mobilidade dos nós, interferência na recepção de sinais e *path loss* (média e variação temporal).
- modelo de radio avançado, baseado em radios reais de comunicação de curto alcance. Permite probabilidade de recepção baseada em SINR (*Signal to INterference Ratio*); múltiplos níveis de energia na transferência com variação individual dos nós; estados (TX, RX, *idle*) com diferentes consumos de energia e atraso.
- modelo de processo físico flexível podendo representar diversos processos, como temperatura, pressão, luz e aceleração.
- modelo de dispositivo de sensoriamento representando barulho, consumo de energia e densidade variável (*bias*).
- *node clock drift*.
- protocolos de roteamento e MAC disponíveis.
- projetado para adaptação e expansão.

Castalia é escrito em C++ e possui um arquivo de configuração de simulação, o `netpp.ini`.

### 2.3.5 NS-2

NS-2 ([UNIVERSITYBERKELEY et al., 2006](#); [DOWNARD, 2004](#)) (*Network Simulator*) é um simulador no nível de aplicação baseado em eventos discretos orientado a objetos, que oferece grande potencial para simular protocolos de roteamento e medições de consumo de energia com diferentes configurações e topologias. NS-2 possui duas linguagens de programação. Foi escrito em C++ e provê uma interface de simulação através da OTcl, um dialeto orientado a objetos do Tcl. O usuário configura e escolhe os parâmetros por meio scripts OTcl, como topologia de rede e fenômenos, e então o programa NS principal faz a simulação com os parâmetros especificados.

### 2.3.6 J-Sim

J-Sim ([SOBEIH et al., 2005](#)), formalmente conhecido como JavaSim, é um ambiente de simulação baseado em componente e de código aberto, escrito em Java. J-Sim foi construído utilizando a arquitetura de componentes autônomos e provê uma definição orientada a objetos. É um simulador de protocolos de RSSF, equivalente ao NS-2.

J-Sim provê um bom modelo de energia e possui suporte para simulação de aplicações e para conexão de sensores reais na simulação (emulação). Para criar, editar e executar modelos, pode-se utilizar o gEditor, que mantém cada um dos modelos de simulação em um arquivo modelo. Um arquivo modelo é um arquivo XML.

### 2.3.7 GloMoSim

GloMoSim ([ZENG; BAGRODIA; GERLA, 1998](#)) é um ambiente de simulação no nível de aplicação, construído para redes móveis sem fio. É escrito em Parsec, que é uma extensão da linguagem C para programação paralela. Tirando vantagem da paralelização, foi mostrado que pode escalar até dez mil nós ([TAKAI \*et al.\*, 2001](#)). GloMoSim parou de ser atualizado e em seu lugar, foi criado um novo produto comercial, QualNet ([TECHNOLOGIES, 2006](#)) que possui pacote específico para RSSF.

### 2.3.8 ATEMU

ATEMU ([POLLEY \*et al.\*, 2004](#)) (*ATmel EMUlator*) é o emulador do processador AVR ([ATMEL, 1997](#)) e da plataforma de sensoriamento MICA2 ([CROSSBOW-TECHNOLOGY, 2002](#)). Suporta apenas o modelo de propagação *free space*. Nós sensores podem executar diferentes códigos ao mesmo tempo. Possui restrições de escalabilidade executando com eficácia aproximadamente até cento e vinte nós. ATEMU é escrito em C e possui um arquivo de especificação de configuração escrito em XML.

### 2.3.9 AVrora

AVrora ([TITZER; LEE; PALSBERG, 2005](#)) é um emulador mais eficiente do que o ATEMU e com corretude parecida com o TOSSIM, segundo [Curren \(2005\)](#). AVrora executa o código instrução-por-instrução, mas melhora a escalabilidade não realizando sincronização dos nós após cada instrução. É implementado em Java.

### 2.3.10 EmStar

EmStar (ELSON *et al.*, 2003) é um simulador baseada no Linux. EmStar pode executar simulação pura e simulação híbrida, que combina simulação com comunicação real com os sensores situados no ambiente, utilizando o mesmo código e arquivos de configuração para executar ambas. O arquivo de configuração deve casar com as configurações de hardware. O modelo de simulação utiliza eventos discretos e é baseada em componentes.

### 2.3.11 OPNET

OPNET (CHANG, 1999) (*OPtimized Network Engineering Tools*) é um simulador comercial, voltado para simulação de redes em geral. Simula redes no nível de aplicação, utilizando modelo hierárquico para montar o sistema. OPNET pode modelar vários tipos de hardware, como plataformas específicas e antenas. O desenvolvimento de modelos pode ser efetuado através de interfaces, e a saída dos resultados pode ser exibida de forma animada.

### 2.3.12 SENS

SENS (SUNDRESH; KIM; AGHA, 2004) (*Sensor, Environment and Network Simulator*) é um simulador para aplicações de RSSF, no nível de aplicação. Tem arquitetura modular e em camadas, com componentes customizáveis que modelam aplicação, comunicação de rede e ambiente físico. O mecanismo de modelagem do ambiente físico tem grande ênfase nas simulações do SENS. Atualmente, existem implementações para concreto, pasto e parede, cada uma delas com suas diferentes características de sinais de propagação. Na verdade, a estrutura funciona como um grafo de componentes, onde o

usuário escolhe suas aplicações, ambientes físicos e tipo de comunicação que melhor lhe convém, interagindo com um componente de ambiente.

### 2.3.13 SENSE

SENSE (CHEN *et al.*, 2004) (*SEnsor Network Simulator and Emulator*) é um simulador desenvolvido especificamente para simular RSSFs no nível de aplicação, que também utiliza a simulação baseada em componentes. SENSE inclui suporte para paralelização da simulação. A principal preocupação dos desenvolvedores na implementação do SENSE é o que consideram os fatores mais críticos de uma ferramenta de simulação: extensibilidade, reusabilidade e escalabilidade. O nó sensor é uma composição de componentes. Consiste de um número menor de componentes primitivos, cada um implementando certa funcionalidade. Conexões entre cada componente são realizadas por *inports* e *outports*, permitindo independência entre os componentes. SENSE objetiva implementar as mesmas funcionalidades do NS-2, mas com uma melhoria no modelo de compartilhamento de pacotes, o que acarreta melhor escalabilidade pela redução de uso de memória. Contudo, o modelo gerado é simples e também possui limitações de comunicação.

A Tabela 2.1 sumariza os objetivos de simulação de cada um dos simuladores descritos, mostrando o que eles permitem simular. Na tabela, aparecem somente os simuladores que ainda estão sendo atualizados por seus desenvolvedores.

## 2.4 Técnicas de Transformação de Modelos

Transformação de modelos é uma abordagem sistemática de alteração da estrutura dos modelos (FENG, 2009). A idéia por trás de transformação de modelos está relacionada

TABELA 2.1 – Modelos que as ferramentas permitem simular

Modelo \ Simulador	TOSSIM	COOJA	OMNeT++	Castalia	NS-2	J-Sim	ATEMU	AVrora	EmStar	OPNET	SENS
Meio físico											
Canal de comunicação											
Rádio											
Sensoriamento											
Bateria											
Emulação											
Redes larga escala											
Implantação de código											
Sistema Operacional											
Aplicação											
Movimento dos nós											
Registro de CPU e memória											
Fenômeno Físico											
Aplicações Heterogêneas											
Suporte Gráfico											

■ - Permite Simular

□ - Não Permite Simular

■ - Permite Simular Parcialmente

com técnicas de compilação, se o modelo for considerado como um programa. Definir transformação de modelos é descrever regras de transformação entre o modelo fonte e o modelo alvo ([WIMMER \*et al.\*, 2007](#)).

[Czarnecki e Helsen \(2003\)](#) classificaram a transformação de modelos em duas grandes categorias: transformação de modelo para código e transformação de modelo para modelo.

Na categoria modelo para código, se distinguem duas abordagens: baseada em visita-ção e baseada em *template*. A abordagem baseada em visitação consiste em prover algum mecanismo de visita para atravessar a representação interna de um modelo e escrever o código em uma estrutura de texto. Uma ferramenta que utiliza essa abordagem é Jamda ([BOOCOCP, 2003](#)). A abordagem baseada em *template* usualmente consiste em um texto alvo contendo meta-código para acessar informações do modelo fonte e fazer a seleção de código, expandindo de maneira iterativa. OptimalJ ([WIKIPEDIA, 2006](#)) e XDE ([RATIONAL, 2006](#)) são ferramentas que utilizam essa abordagem.

A categoria modelo para modelo traduz modelos fonte para modelos alvo. Essa abordagem facilita a transformação quando existem grandes saltos de abstração entre fonte e alvo, gerando modelos intermediários, o que torna a transformação mais modular ([CZARNECKI; HELSEN, 2003](#)). Como exemplo dessa categoria, tem-se a transformação baseada em grafos. Transformação baseada em grafos é especificada por regras de transformação. Aplicação dessas regras possuem dois passos: localização do sub-grafo no grafo representando o modelo, e transformação desse sub-grafo para obter o resultado. Como exemplo dessa abordagem tem-se VIATRA ([VARRÓ; VARRÓ; PATARICZA, 2002](#)), ATOM ([LARA; VANGHELUWE, 2002](#)) e GreAT ([AGRAWAL; KARSAI; SHI, 2003](#)).

Outro exemplo de transformação de modelo para modelo é XSLT ([W3C, 1998](#)) (Xml Stylesheet Language Transformations), que é a tecnologia padrão para transformação de



documentos XML ([CONSORTIUM, 1998](#)). XML é uma linguagem para a troca de dados entre aplicações e tem se mostrado amplamente aceita como especificação de interfaces comuns entre diferentes ferramentas. Adicionalmente, a XML possibilita a extensão do conjunto de *tags*, permitindo que novas ferramentas sejam adicionadas futuramente, alterando a estrutura padrão para troca de dados.

XSLT define a sintaxe e a semântica dos documentos de transformação, o que inclui regras declarativas para a transformação de uma árvore XML em outra.

Uma folha de estilo XSLT é um conjunto de regras. Cada regra em uma folha de estilo é dividida em duas partes: uma expressão Xpath que seleciona quais nodos da árvore a regra se aplica e o corpo da regra ([ARCINIEGAS, 2001](#)). Um processador XSLT verifica o documento de origem e aplica as regras gerando outro documento. Esse processo é ilustrado na Figura 2.5.

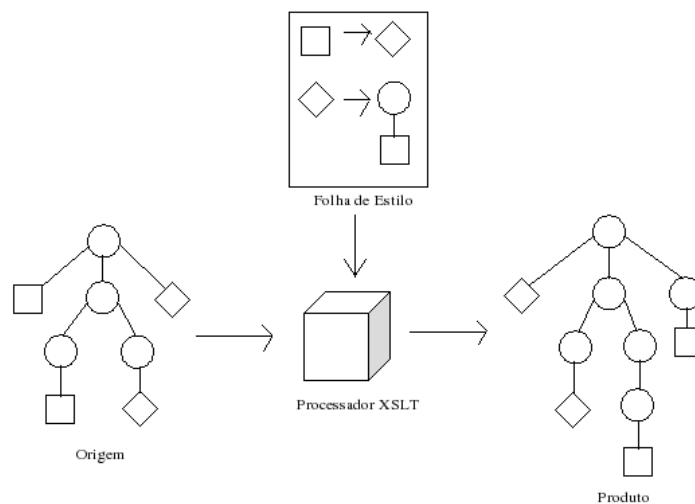


FIGURA 2.5 – Visão Geral do Processo XSLT

XSLT também pode ser incluída na categoria de transformação de modelo para código, pois, além de transformar um documento XML em outro documento XML, também pode transformar um documento XML em um documento texto, gerando código diretamente.

Essa é a abordagem utilizada neste trabalho.

## 2.5 Trabalhos Relacionados

Várias pesquisas têm sido realizadas objetivando encontrar mecanismos que possibilitem a integração de ferramentas de simulação, seja na fase de modelagem ou na fase de implementação. Esta seção apresenta os principais trabalhos da literatura relacionados com o contexto desta tese.

[Sridharan, Zuniga e Krishnamachari \(2004\)](#) integraram o simulador de ambiente MATLAB com o simulador do TinyOS, o TOSSIM. O objetivo era realizar simulação de rede com dados de sensoriamento coletados de maneira interativa. Foi utilizado o TinyViz ([LEVIS \*et al.\*, 2003](#)), que é a ferramenta de visualização do TOSSIM, como intermediador entre os dois simuladores. O MATLAB produzia amostras, que eram enviadas como eventos para o TinyViz, que, por sua vez, inseria esses eventos na fila de eventos do TOSSIM. [Cheong, Lee e Zhao \(2006\)](#) criaram o Viptos (Visual Ptolomy and TinyOS) que é uma ferramenta para integração do TOSSIM e do Ptolomy II ([EKER \*et al.\*, 2003](#)), um ambiente gráfico de modelagem e simulação de sistemas embarcados. O usuário faz a modelagem utilizando o ambiente gráfico e o código TinyOS é criado automaticamente. O intuito dos criadores do VIPTOS é permitir que usuários transitem facilmente de modelagem de alto nível para simulação de código real e implantação de RSSF. Esses dois trabalhos apresentam formas de integração que são dependentes das ferramentas. Ambos os trabalhos integraram duas ferramentas específicas para atender objetivos pontuais. Em contraste, o presente trabalho objetiva uma forma de integração ampla, sem procurar concentrar-se em simuladores específicos.

Dulman, Kaya e Koprnikov (2005) e Girod *et al.* (2004) criaram formas de permitir que usuários executem aplicações TinyOS diretamente em suas ferramentas de simulação. Dulman, Kaya e Koprnikov (2005) criou NesCT, que faz com que aplicações TinyOS sejam executadas diretamente no OMNeT++. Girod *et al.* (2004) criou o EmTOS, que provê essa mesma funcionalidade para o EmStar. O objetivo desses dois trabalhos é permitir a execução de aplicações TinyOS em seus simuladores, e não uma integração geral entre simuladores, conforme o presente trabalho propõe.

Vitorino *et al.* (2004) e Olivieri (2009) propõem ferramentas onde, a partir de especificações, os códigos-fonte de uma aplicação são gerados automaticamente para plataformas de RSSFs específicas, escolhidas pelo usuário. Vitorino *et al.* (2004) criou o WISDOM, uma ferramenta visual capaz de gerar código nativo para uma dada plataforma computacional. Os objetivos do WISDOM são simplificar o desenvolvimento de aplicações para RSSFs e prover um modelo de programação multiplataforma, ou seja, que permita que uma aplicação possa ser especificada e codificada uma única vez para diversas plataformas computacionais. Atualmente WISDOM suporta as seguintes plataformas: TinyOS/*Mica* *Motes* e Yatos/BEAN. Ela foi implementada na linguagem Java, e possui código aberto. Olivieri (2009) fez o protótipo de um arcabouço para modelar, simular e gerar código para aplicações RSSF baseadas na ferramenta MathWorks (MATHWORKS, 1994). A idéia é que o usuário se concentre na modelagem e a geração e execução do código seja feita de forma automática para diferentes plataformas. O protótipo tem o objetivo de gerar código para os sistemas operacionais TinyOS e MANTIS. Foi implementado na linguagem C. O objetivo dos dois trabalhos mostrados é fazer modelagem unificada de plataformas de RSSFs, onde os códigos específicos de cada plataforma são gerados automaticamente.

Utilizando a tecnologia GME – Generic Modeling Environment (LEDECZI *et al.*,

2001), foi desenvolvido MILAN ([BAKSHI; PRASANNA; LEDECZI, 2001](#)), um arcabouço de simulação para projeto e otimização de sistemas embarcados, através da integração de simuladores existentes amplamente aceitos e utilizados. MILAN provê um paradigma formal para especificação estrutural e comportamental de aspectos de sistemas embarcados. Dentre os simuladores integrados no MILAN tem-se: MATLAB ([MATHWORKS, 1984](#)), SimpleScalar ([BURGER; AUSTIN, 1997](#)) e SystemC ([SYSTEMC, 1999](#)). Contudo, esses simuladores não são voltados para o estudo de RSSFs.

Os trabalhos mostrados apresentam formas de integração que são dependentes das ferramentas. Normalmente, procurou-se integrar duas ferramentas específicas para atender objetivos pontuais. O último trabalho citado ([LEDECZI \*et al.\*, 2001](#)) objetiva uma integração mais geral, contudo não é voltado para simuladores de RSSF. Em contraste, o presente trabalho propõe uma forma ampla de integração, na fase de modelagem da simulação, que pode incluir diversos simuladores em todos os níveis de operação: simuladores de aplicação, simuladores de sistema operacional e simuladores de nível de instrução.

## 2.6 Sumário

Neste capítulo foram apresentados os principais conceitos e terminologias relacionadas com o contexto desta tese e que também a fundamentam. Primeiramente, foram mostrados alguns aspectos importantes de simulação discreta, RSSFs e as principais características dos simuladores de RSSFs. Em seguida, foram abordadas algumas técnicas de transformação de modelos, procurando justificar as escolhas utilizadas no presente trabalho.

Foram apresentados alguns trabalhos relacionados que mostram diferentes formas de

---

integração. Contudo, as formas de integração mostradas são dependentes das ferramentas envolvidas, pois são voltadas para a integração de simuladores específicos. Em contraste, o presente trabalho propõe uma forma de integração generalizada, que pode incluir diversos tipos de simuladores.

# 3 Modelo de Dados de Simulação de RSSF

RSSF são um tipo de tecnologia que realiza monitoramento através da coleta de dados diretamente no ambiente onde são instaladas. O desenvolvimento de aplicações RSSF é uma tarefa complexa, pois existem várias restrições de fatores, como por exemplo:

- limitações de recursos físicos como bateria, alcance das antenas, memória de armazenamento e poder de processamento;
- possibilidade de implantação da rede em ambiente hostil, como no mar, vulcão ou floresta, sujeito a condições climáticas severas, o que pode resultar em falha nos equipamentos e interferências no sensoriamento e na comunicação.
- forte dependência dos requisitos de aplicação que normalmente incluem tolerância a falhas, escalabilidade, baixo custo, eficiência no consumo de energia, modo de operação autônoma, flexibilidade e privacidade (ILYAS; MAHGOUB, 2005);
- Falta de uma maneira organizada para o desenvolvimento de aplicações RSSF formalmente estabelecida.

Conforme mencionado por [Egea-Lopez et al. \(2005\)](#), [Levis et al. \(2003\)](#), simulação é uma importante ferramenta no desenvolvimento de sistemas para RSSFs. A necessidade de realizar simulação em diversas ferramentas para combinar resultados de simulação complementares, foi a motivação para a criação de um modelo de dados abstrato para configuração de simulação, que possa ser compartilhado por diversos simuladores. A partir de agora, chamaremos de MD o Modelo de Dados para a configuração da simulação de RSSF proposto.

Este capítulo está organizado da seguinte forma. A Seção 3.1 define o MD. A Seção 3.2 descreve e apresenta as características do MD. A Seção 3.3 mostra como utilizar o MD. A Seção 3.4 mostra como são efetuadas as transformações no MD para se chegar em um modelo específico de simulação. Por fim, a Seção 3.5 apresenta as considerações finais deste capítulo destacando as principais contribuições do MD e suas limitações.

### 3.1 Definição do MD

Um projeto de simulação se inicia com uma descrição adequada de uma aplicação real. Essa descrição constitui o modelo de simulação, construído a partir de conceitos comuns de simulação, como entidades, atributos e eventos. São declaradas as estruturas da simulação em termos de entidades e são implementados seus comportamentos.

Pacotes comuns de simulação claramente separam implementação de descrição do modelo e instanciação ([EGEA-LOPEZ et al., 2005](#)). O núcleo da simulação, como bibliotecas de simulação, é escrito em linguagem de alto nível, como C++ e Java e a descrição do modelo normalmente é efetuada por alguma linguagem *script* como Tcl, Perl e Python ou de marcação, como XML. Esse fato pode ser observado nos exemplos de simuladores

mostrados na Seção 2.3. *Scripts* proporcionam uma abordagem eficiente para descrição e configuração de modelos e instanciação de execução de simulação. Desta forma, ferramentas de simulação usualmente consistem de uma biblioteca básica de simulação e alguma linguagem *script* para descrição da simulação.

Por esta razão, o trabalho concentra-se na criação de um arquivo comum de configuração da simulação, onde são colocados os parâmetros relevantes de simulação. MD é, na verdade, um arquivo de configuração de simulação. Cenários de simulação descritos em um arquivo de configuração comum podem ser utilizados em diversos simuladores. Essa proposta é mostrada na Figura 3.1, onde o MD, que é o arquivo de configuração comum, é convertido, através de regras de transformação, para o arquivo de configuração dos simuladores. Essa abordagem de integração permite utilizar simuladores que provêm algum tipo de arquivo de configuração de simulação e independe do tipo de simulador, seja ele de aplicação, sistema operacional ou instrução.

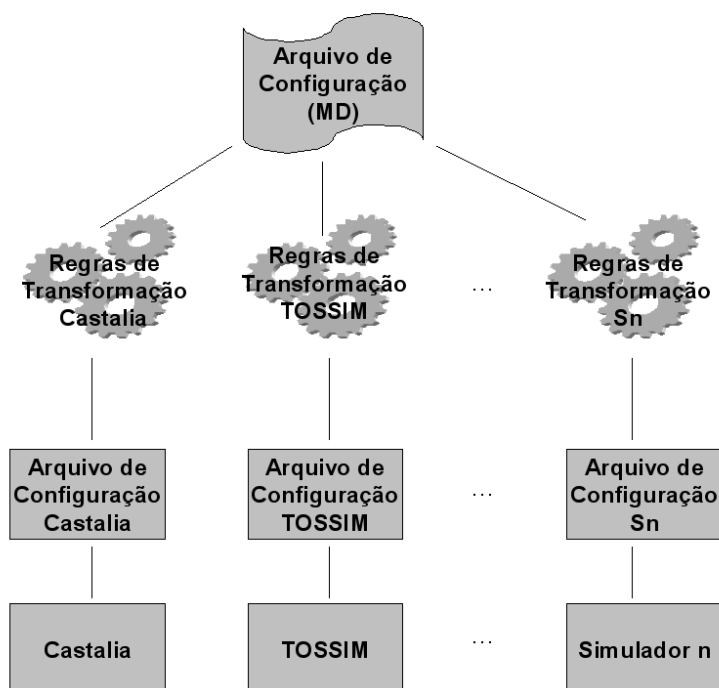


FIGURA 3.1 – Proposta do modelo de dados extensível



Como não há um consenso sobre a linguagem utilizada nos arquivos de configuração dos simuladores (como exemplos tem-se Tcl, Python, XML e Perl), optou-se por utilizar a linguagem XML para montar o MD, pois é uma linguagem para troca de dados, que permite facilmente realizar extensões e alterações, possibilitando que novos simuladores possam ser adicionados futuramente.

## 3.2 Descrição do MD

O MD é um arquivo de configuração da simulação. Isso significa que ele deve conter a descrição completa da simulação. A descrição da simulação é armazenada no MD na forma de parâmetros e seus respectivos valores.

Como foi visto na Seção 2.1, no processo de desenvolvimento de um estudo de simulação, os primeiros passos são definição dos objetivos e requisitos, seguido pela coleta de dados. Esses são os passos necessários a todo e qualquer projeto de simulação, independentemente de plataforma ou simulador que será utilizado. Então os primeiros parâmetros de descrição da simulação que devem estar contidos no MD dizem respeito aos objetivos e requisitos da simulação que se deseja executar. Os objetivos de simulação devem ser explicitamente definidos, assim como sua profundidade e amplitude. Os parâmetros incluídos no MD que dizem respeito aos requisitos e objetivos de simulação são mostrados, resumidamente na Tabela 3.1.

RSSFs pertencem à área de pesquisa aplicada; então, para se chegar a resultados satisfatórios de simulação, é necessário fazer uma descrição detalhada do cenário da aplicação, para que o modelo se torne o mais próximo possível da realidade.

Um grande desafio para a construção do MD é a quantidade de parâmetros que devem

TABELA 3.1 – Parâmetros contidos no MD

Cenário	Pontos de Destaque	Descrição
O que será monitorado	Tipo de sensor	Qual fenômeno físico será monitorado, usando que tipo de sensor. É mais de um fenômeno? Quantos?
	Consumo de energia	Quanto o sensoriamento consome. Por exemplo, o sensor de umidade consomem mais que o sensor de temperatura. Também é importante comparar com o consumo do rádio.
	Alcance	Especificar o raio de alcance do sensor.
Topologia	Quantidade de sensores	Depende da aplicação e pode ser modificado de acordo com os resultados da simulação.
	Região de distribuição dos sensores	Pode ser em ambiente fechado, como monitoramento industrial ou em uma floresta ou vulcão.
	Tamanho da região de distribuição	Quais as dimensões aproximadas da distribuição dos sensores.
	Modo de distribuição	Os sensores serão distribuídos randomicamente ou de maneira uniforme, em lugares pré-determinados. Depende da cobertura que se deseja alcançar.
	Nó Usuário ou observador	Se tiver, quantos são. Devem ficar em local pré-determinado?
Tráfego de dados esperado	Frequência de sensoriamento	Pode variar amplamente, dependendo da aplicação. Deve ser o mínimo necessário e suficiente para propiciar economia de energia.
	Frequência de envio de dados ao nó usuário	Se o que está sendo monitorado não é crítico, pode levar até horas para o envio de dados ao nó usuário. Pode acontecer também de só enviar dados se ocorrer algum evento determinado.
	Tamanho dos pacotes de dados	Depende do que se quer analisar.
	Fidelidade de dados	Pode haver perda de dados ou todas as amostras de dados têm que necessariamente chegar ao destino?
	Frequência de envio de estímulos	Frequência que o fenômeno envia os estímulos ao canal de sensoriamento.
Métricas de Avaliação	Tempo de vida	A rede precisa ter um tempo de duração mínimo? Definir o que é vida útil para a rede.
	Consumo de energia	Se for necessário, definir a granularidade.
	Escalabilidade	Quantos nós o simulador consegue simular?
	Latência	Diferença de tempo entre envio e recepção.
	Perda de pacotes	Porcentagem de pacotes perdidos.
	Validação de código	Verifica se o código executado está ok.

ser disponibilizados, de modo que possa incluir o grande leque de aplicações que uma RSSF pode cobrir. Os parâmetros incluídos no MD foram baseados nos simuladores estudados na Seção 2.3 e também no modelo de dados mostrado na Seção 2.2.1, onde foram identificados os elementos comuns de simulação de uma RSSF.

Dessa forma, o modelo de dados foi construído como se fosse uma árvore, onde a raiz é a rede que se deseja simular, a qual contém quatro filhos: tempo de simulação, topologia, resultados de simulação e nós. Esses são alguns dos elementos comuns identificados no modelo de simulação de RSSF. Cada um desses filhos pode conter outros filhos ou atributos, e assim por diante, até chegar nas folhas. Os atributos são os parâmetros que contêm efetivamente os valores que são utilizados na simulação. As folhas podem não ser elementos comuns dos simuladores. Nessa etapa, já aparecem os elementos específicos de cada simulador. Uma ilustração, contendo os três primeiros níveis que representam os principais elementos da árvore de simulação constante no MD é mostrada na Figura 3.2. Nas caixas aparecem os nós da árvore e embaixo dos nós aparecem seus atributos.

Nesse primeiro momento, o MD não engloba todos os parâmetros possíveis de todas as ferramentas de simulação de RSSF. Procurou-se inserir, de forma incremental, os parâmetros de simulação requeridos para simular corretamente uma aplicação de RSSF. Os parâmetros podem ser facilmente incluídos no MD, caso seja necessário. Os simuladores que não suportam alguns parâmetros específicos, podem ignorá-los durante seu processo de modelagem de cenário.

Um exemplo de como são inseridos os valores no MD é apresentado na Figura 3.3. É mostrada a descrição da topologia de uma RSSF com 15 nós sensores, distribuídos de maneira uniforme em um terreno de dimensões 50m por 50m.

O arquivo *XML Schema* completo com o MD está disponível no Anexo A. Nele podem

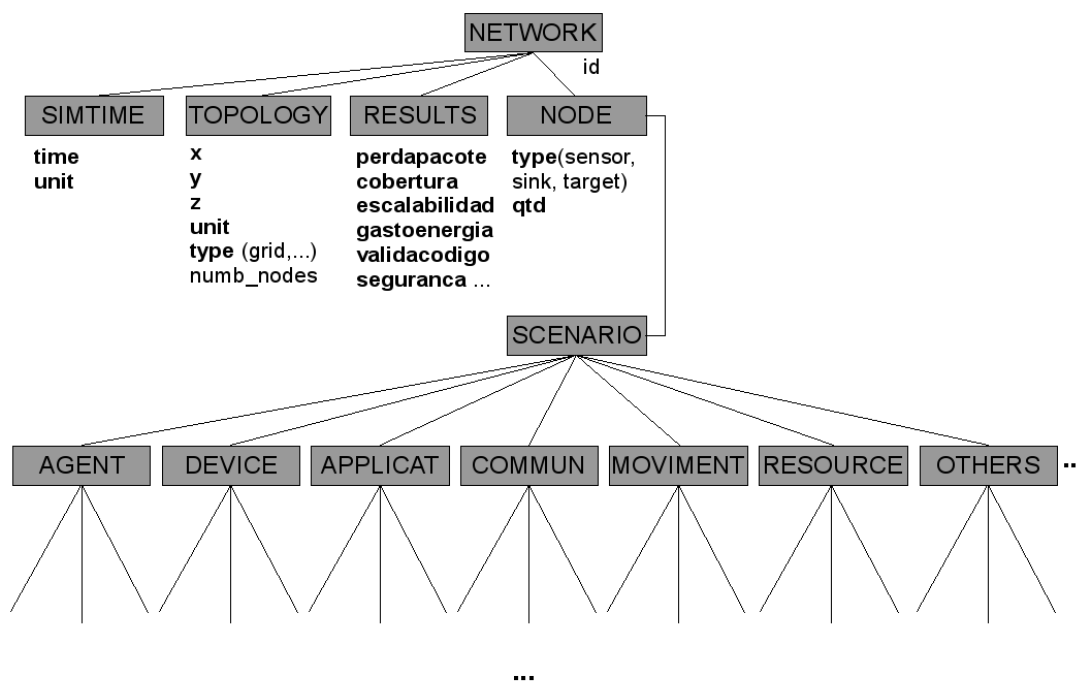


FIGURA 3.2 – Ilustração do MD como árvore XML

ser verificados todos os parâmetros disponibilizados para compor a simulação.

```

<Topology.
.   Number_of_nodes="15".
.   unit="metros".
.   EnvironmentDimensionX="50".
.   EnvironmentDimensionY="50".
.   EnvironmentDimensionZ="0">
.   <Type name="Uniform"/>
</Topology>

```

FIGURA 3.3 – Exemplo de descrição da topologia no MD

### 3.3 Utilização do MD

O MD promove a viabilização do compartilhamento de modelos de configuração da simulação por diversos simuladores. Quando for necessário alterar algum parâmetro de simulação, este será alterado diretamente no MD, o que faz com que a alteração ocorra automaticamente para todos os simuladores que utilizam este parâmetro.

Os passos para executar a simulação utilizando o MD são, basicamente:

1. O arquivo vai sendo construído de acordo com os requisitos, objetivos e restrições da aplicação que se deseja simular.
2. Tendo como base o item anterior, o simulador é escolhido e os dados que esse simulador necessita como entrada também são incluídos no MD.
3. O MD então é transformado no arquivo de simulação do simulador escolhido, através de regras.
4. A simulação é executada. Caso seja necessário fazer alterações, volta-se ao item 1. O processo é repetido até que se chegue aos resultados esperados.
5. Uma vez que o MD está validado, pode-se utilizá-lo como entrada para outro simulador.

Existem dois tipos de parâmetros incluídos no MD: requisitos da aplicação, que permanecem fixos, e dados específicos de entrada de simuladores, que variam de acordo com o simulador que se está utilizando, pois os simuladores normalmente têm necessidades diferentes de dados de entrada. Para cada simulador que se deseja utilizar, são incluídos os parâmetros que ainda não estão contemplados. Atualmente, o MD inclui os principais requisitos e objetivos de uma aplicação RSSF e os dados de entrada necessários para os simuladores Castalia e TOSSIM.

Dessa forma, MD possui os seguintes requisitos:

- é extensível. Essa é apenas a primeira proposta e não a última;
- modela os requisitos não funcionais da aplicação, que são as configurações gerais da simulação de RSSF, ou seja, as características comuns da aplicação, descritas de forma independente, sem tomar conhecimento de ferramentas de simulação;

- modela os dados de entrada do simulador, escolhido a partir dos requisitos e restrições;
- todas as alterações de configuração da simulação são efetuadas diretamente no MD, que serve de modelo extensível de dados.

### 3.4 Transformação do MD em Modelo de Simulação

Uma vez que o MD está montado, ele precisa ser transformado no arquivo de configuração do simulador escolhido para que a simulação possa ser realizada. Esse processo pode ser efetuado utilizando-se alguma técnica de transformação de modelos, conforme mencionado na Seção 2.4. No modelo proposto, essa transformação é realizada aplicando-se folhas de estilo da linguagem de transformação do XML, que é a XSLT. XSLT define a sintaxe e a semântica dos documentos de transformação, o que inclui regras declarativas para a transformação de um arquivo XML em outro arquivo.

Os arquivos de configuração de cada simulador são programas capazes de realizar simulação que, por sua vez, são arquivos texto, escritos em alguma linguagem *script* ou de alto nível. XSLT permite que arquivos XML sejam transformados em outros arquivos XML, HTML ou texto, conforme ilustrado na Figura 3.4. A saída no formato texto é a utilizada, por ser a que melhor se adequa ao presente trabalho.

Na folha de estilos estão contidas regras que apontam o tipo de arquivo que deve ser criado e toda a parte textual pertencente a este arquivo, como comandos e comentários. Em seguida, cada nó da árvore XML no arquivo-fonte vai sendo percorrido, e seus respectivos valores vão sendo preenchidos no arquivo-alvo. Ao final, tem-se o arquivo de configuração criado, pronto para ser compilado. Maiores detalhes das regras de transfor-

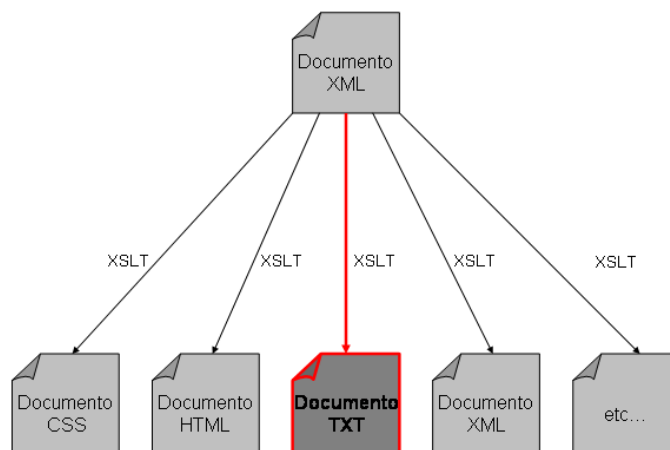


FIGURA 3.4 – Utilização da XSLT para transformar arquivos XML em uma variedade de formatos

mação são apresentadas nas Seções 4.1 e 4.2. Deve ser criada uma folha de estilos para cada simulador incluído no MD.

### 3.5 Sumário

Neste capítulo foi descrito o MD - modelo de dados unificado para a configuração de simulação de RSSF. MD apresenta-se como solução para realizar modelagem abstrata no desenvolvimento de aplicações de RSSF.

MD foi projetado para ser um modelo extensível dos dados de simulação, de forma a ser compartilhado por vários simuladores de RSSF. Tem como objetivo englobar os parâmetros de simulação requeridos para simular corretamente uma RSSF e dados específicos de entrada de simuladores.

A principal vantagem do MD é a utilização da mesma configuração de simulação por mais de um simulador. Quando for necessário alterar algum parâmetro de simulação, este

será alterado diretamente no MD, o que faz com que a alteração ocorra automaticamente para todos os simuladores que estão utilizando o parâmetro.

A atual versão do MD apresenta algumas limitações. Ela é útil para simuladores que provêm algum tipo de arquivo de configuração de simulação. Se o código fonte do simulador precisar ser manipulado para realizar configuração de simulação, o MD não permite a geração do modelo. Existem dois simuladores contemplados na versão atual, TOSSIM e Castalia. No próximo capítulo será mostrado um exemplo de uso do MD para esses simuladores.



## 4 Exemplo de Uso

Para que o MD tenha validade, é necessário que ele possa ser transformado em um modelo de simulação de uma ferramenta. Além do mais, como a proposta é compartilhar o MD pelos simuladores, então a transformação terá que ser efetuada para pelo menos dois simuladores.

Para exemplificar o uso do MD, foram escolhidos os simuladores Castalia e TOSSIM. Castalia é um simulador no nível de aplicação, independente de plataforma, desenvolvido especialmente para a simulação de RSSF. Castalia é baseado no OMNeT++, um dos mais populares simuladores de redes genéricas. TOSSIM é o simulador do sistema operacional TinyOS, que está se tornando o SO padrão de RSSF, segundo [Egea-Lopez \*et al.\* \(2005\)](#). Descrições mais detalhadas dos dois simuladores encontram-se no Capítulo 2. Ambos possuem arquivo de configuração de simulação e objetivos de simulação distintos, o que faz com que haja maior benefício na execução de simulação utilizando esses dois simuladores, pois os resultados de simulação serão complementares.

Este capítulo está organizado da seguinte forma. A Seção 4.1 mostra como foram aplicadas as regras de transformação no Castalia. A Seção 4.2 mostra como foram aplicadas as regras de transformação no TOSSIM. E a Seção 4.3 apresenta as considerações finais deste capítulo.

## 4.1 Aplicação com Castalia

O Castalia separa implementação de configuração da simulação. A implementação é efetuada utilizando a linguagem NED que define os módulos, sub-módulos e interfaces. O comportamento do módulo é efetuado utilizando a linguagem C++. Normalmente o usuário não altera esses arquivos (BOULIS, 2009). A configuração da simulação é efetuada utilizando-se o arquivo `omnetpp.ini`, onde são definidos todos os parâmetros usados pelo OMNeT e Castalia. Esse é o arquivo criado a partir do MD.

No arquivo `omnetpp.ini` é possível configurar por volta de 90 parâmetros que estão associados a cada módulo do Castalia: processo físico, topologia, equipamento de sensoriamento, canal sem fio, radio, camada MAC, roteamento, dentre outros. Esses parâmetros são alimentados a partir do que foi configurado no MD, usando as regras de transformação constantes na folha de estilos criada para o Castalia.

Um exemplo das regras criadas para o Castalia é mostrado na Figura 4.1. Aí são mostradas as regras que fazem o tratamento da topologia e são inseridas no `omnetpp.ini`. Se essas regras forem aplicadas na descrição da topologia no MD mostrada na Figura 3.3 do capítulo anterior, o resultado ficaria como o mostrado na Figura 4.2. Foram retiradas do MD as dimensões X e Y, o número de nós e a topologia do tipo uniforme, representada no Castalia pelo número 0. Essas são apenas parte das regras que constam na folha de estilos criada para o Castalia. As regras vão sendo executadas em todo o arquivo MD, até que se chegue ao arquivo `omnetpp.ini` completo.

A lista completa de parâmetros existentes para configuração da simulação no Castalia pode ser encontrada no Anexo A.

Para realizar a configuração de diversos módulos no Castalia, foi criado um repositório

```

# COMPOUND MODULE: SN (the network)
</xsl:text>
# define deployment details
SN.field_x = <xsl:value-of select="Topology/@EnvironmentDimensionX" />
SN.field_y = <xsl:value-of select="Topology/@EnvironmentDimensionY" />
SN.numNodes = <xsl:value-of select="Topology/@Number_of_nodes" />
<xsl:choose>
<xsl:when test="Topology/Type/@name='Grid'">
<xsl:text>
SN.deploymentType = 1

include ../Parameter_Include_Files/BackwardCompatibility.ini
</xsl:text>
</xsl:when>
<xsl:when test="Topology/Type/@name='Uniform'">
<xsl:text>
SN.deploymentType = 0

include ../Parameter_Include_Files/BackwardCompatibility.ini
</xsl:text>
</xsl:when>
<xsl:when test="Topology/Type/@name='Random'">
<xsl:text>
SN.deploymentType = 2

include ../Parameter_Include_Files/BackwardCompatibility.ini
</xsl:text>
</xsl:when>
<xsl:when test="Topology/Type/@name='Other'">
SN.deploymentType = 3
include node_locations.ini
</xsl:when>
</xsl:choose>

```

FIGURA 4.1 – Regras para tratar a topologia no Castalia

```

# COMPOUND MODULE: SN (the network)

# define deployment details
SN.field_x = 50
SN.field_y = 50
SN.numNodes = 15
SN.deploymentType = 0

include ../Parameter_Include_Files/BackwardCompatibility.ini

```

FIGURA 4.2 – Resultado após aplicar regras de transformação

de arquivos '.ini' em um diretório específico. Esses arquivos são incluídos no omnetpp.ini. Essa separação foi feita para facilitar a configuração e para que o omnetpp.ini não ficasse muito carregado. Assim, alguns parâmetros existentes no MD precisam ser passados para esses arquivos e não diretamente para o omnetpp.ini. As regras de transformação contemplam a criação e chamada desses outros arquivos.

## 4.2 Aplicação com TOSSIM

O TOSSIM é o simulador do sistema operacional TinyOS. Um programa TinyOS consiste de um conjunto de componentes NesC. Um componente realiza alguns tipos de serviços, que são especificados por interfaces. Existem dois tipos de componentes: módulo e configuração. O módulo contém implementação de suas interfaces. A configuração contém a interligação dos componentes utilizados.

Um exemplo de módulo e de configuração pode ser visualizado na Figura 4.3. SenseToLeds é um programa que mostra o valor de um sensor de luz, em binário, nos LEDs de um *mote*. O programa na verdade é um componente TinyOS, que é do tipo configuração. Sendo assim, contém uma interligação de todos os componentes utilizados, que são Main, SenseToInt, IntToLeds, TimerC e DemoSensorC. Também é mostrado o componente SenseToInt, que é do tipo Módulo. Estes são apenas alguns dos componentes disponíveis na biblioteca TinyOS.

```

SenseToLeds {
} implementation { components Main, SenseToInt,
IntToLeds, TimerC,
DemoSensorC as Sensor;
Main.StdControl -> SenseToInt;

Main.StdControl -> IntToLeds;
SenseToInt.Timer -> TimerC.Timer[unique("Timer")];
SenseToInt.TimerControl -> TimerC;
SenseToInt.ADC -> Sensor;
SenseToInt.ADCControl -> Sensor;
SenseToInt.IntOutput -> IntToLeds;
}

module SenseToInt { provides { interface StdControl;
}uses { interface Timer;
interface StdControl
as TimerControl;
interface ADC;
interface StdControl
as ADCControl;
interface IntOutput;
}
} implementation { ...
}

```

FIGURA 4.3 – Exemplos dos dois tipos de componentes TinyOS

TOSSIM separa implementação de configuração da simulação. A implementação é feita utilizando C e NesC. A configuração é feita utilizando-se C++ e Python. Apesar dessa separação, alguns parâmetros de entrada do TOSSIM precisam ser manipulados diretamente no código-fonte, definidos no sistema operacional.

Normalmente os parâmetros que precisam ser modificados no SO são incluídos em um arquivo com extensão .h. Eventualmente, alguns outros parâmetros são incluídos no componente do tipo módulo, na parte de implementação. A folha de estilos do TOSSIM provê a criação de uma espécie de esqueleto desses arquivos, inserindo exclusivamente os parâmetros e valores que constam no MD. Então, além de criar o arquivo de configuração da simulação, a folha de estilos também cria o esqueleto desses componentes. Como exemplo, caso seja necessário inserir na aplicação o parâmetro que indica a frequência de amostragem de leituras para a aplicação OSCILLOSCOPE, a folha de estilos cria o esqueleto do arquivo .h, conforme mostrado na Figura 4.4.

```
#ifndef OSCILLOSCOPE_H
#define OSCILLOSCOPE_H

enum {

    /* Default sampling period. */
    DEFAULT_INTERVAL = 256,

};

#endif
```

FIGURA 4.4 – Esqueleto do arquivo OSCILLOSCOPE.h

A topologia que é desenvolvida no TOSSIM é completamente diferente do que ocorre no Castalia. Para o TOSSIM, a localização física dos nós não é muito importante. Existem apenas dois parâmetros que são levados em consideração: *gain* e *noise*. *Gain* representa a probabilidade que um nó tem de conseguir receber um pacote. *Noise* representa inter-

ferência externa a rede e barulho. Esses valores geralmente são inseridos na configuração da simulação através de arquivos, pois é necessária a definição desses valores para cada ligação entre dois nós da rede. A regra que trata a topologia no TOSSIM é mostrada na Figura 4.5 e o resultado pode ser visualizado na Figura 4.6.

```
#define parameter gain
f = open (<xsl:value-of select="Topology/@GFile"/> "r")
<xsl:text>
lines = f.readlines()
for line in lines:
    s = line.split()
    if (len(s) > 0):
        if (s[0] == "gain"):
            r.add(int(s[1]), int(s[2]), float(s[3]))
</xsl:text>
```

FIGURA 4.5 – Parte da regra para tratar a topologia no TOSSIM

```
f = open("topo.txt", "r")
lines = f.readlines()
for line in lines:
    s = line.split()
    if (len(s) > 0):
        if (s[0] == "gain"):
            r.add(int(s[1]), int(s[2]), float(s[3]))
```

FIGURA 4.6 – Resultado após aplicar regras de transformação

A lista completa de parâmetros existentes para configuração da simulação no TOSSIM pode ser encontrada no Anexo A.

### 4.3 Sumário

Neste capítulo foram mostrados dois exemplos de uso do MD executados nos simuladores Castalia e TOSSIM. Ambos os simuladores apresentam arquivos de configuração de simulação, separados de implementação. O objetivo desses exemplos é analisar se o MD pode efetivamente ser transformado em um arquivo de configuração específico.

No Castalia, a configuração da simulação é realizada no arquivo omnetpp.ini. Foi

criada uma folha de estilos XSLT que apresenta regras para transformar o MD no `omnetpp.ini`. Foi mostrado um exemplo de configuração de topologia efetuada no MD e o tratamento que é efetuado para que seja transformada na configuração de topologia no `omnetpp.ini`.

Para o TOSSIM, foi criada uma outra folha de estilos XSLT com diferentes regras. O MD é então transformado no arquivo de configuração python, existente no TOSSIM. Contudo, alguns parâmetros precisam ser definidos diretamente nos módulos do TinyOS. Sendo assim, um esqueleto dos módulos é criado, contendo apenas os parâmetros e valores que constam no MD e que devem ser configurados no TOSSIM.

O próximo passo é realizar a configuração de uma aplicação ilustrativa para fazer a verificação de que o MD cumpre seu objetivo de ser modelo extensível de dados. Essa aplicação será apresentado no capítulo seguinte.

# 5 Aplicação

Como prova de conceito, será realizada a simulação de uma aplicação ilustrativa usando o MD, apresentado no Capítulo 3. Este capítulo está organizado da seguinte forma. A Seção 5.1 descreve a aplicação desenvolvida para testar o MD. A Seção 5.2 mostra o que será investigado e o ambiente de simulação. A Seção 5.3 apresenta os resultados obtidos e análises. Por fim, a Seção 5.4 apresenta as constatações deste capítulo.

## 5.1 Descrição

A cachaça é uma bebida genuinamente brasileira, conhecida mundialmente. Sua produção teve início no século XVI. Segundo a lei brasileira, que padroniza e classifica bebidas, a cachaça é definida como “a denominação típica e exclusiva da Aguardente de Cana produzida no Brasil, com graduação alcoólica de 38% vol (trinta e oito por cento em volume) a 48% vol (quarenta e oito por cento em volume) a 20<sup>o</sup>C (vinte graus Celsius), obtida pela destilação do mosto fermentado do caldo de cana-de-açúcar com características sensoriais peculiares, podendo ser adicionada de açúcares até 6g/l (seis gramas por litro), expressos em sacarose”.

Segundo [Silva \(2006\)](#), o processo produtivo da cachaça segue, resumidamente, o fluxo apresentado na Figura 5.1.



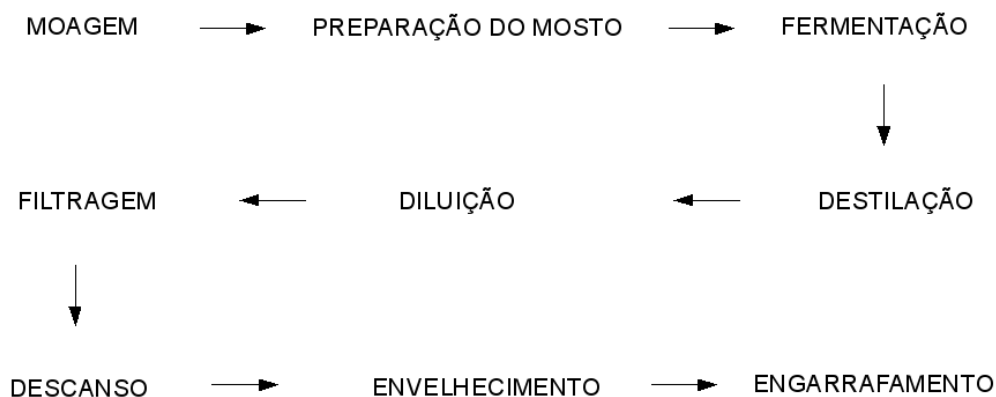


FIGURA 5.1 – Resumo do Fluxo Produtivo da Cachaça

A partir da filtragem e descanso, a cachaça já pode ser engarrafada ou ir para tonéis de madeira para envelhecimento. No processo de envelhecimento, as características sensoriais da cachaça se modificam, aprimorando suas qualidades com novos aromas, novo paladar e nova coloração, tornando-a mais agradável do que a cachaça recém destilada, o que lhe agrega maior valor (SEBRAE-MG, 2001). Quanto maior o período de envelhecimento, maior o valor agregado da bebida (SEBRAE-MG, 2001). Pelos motivos expostos, esta aplicação se concentrará no monitoramento da fase de envelhecimento da cachaça. A simulação do processo de envelhecimento pode ser facilmente estendida para outras bebidas que precisam passar pelo mesmo processo, como o vinho, por exemplo, ou para outras aplicações que necessitam de condições ambientais controladas, seja em monitoramento de bebidas, comidas, medicamentos, e assim por diante.

O descanso, que é a primeira fase de estocagem da cachaça, deve ser de dois a quatro meses, normalmente feito em tanques de alvenaria ou de aço inoxidável (SEBRAE-MG, 2001). A partir dos quatro meses, ela passa para o envelhecimento propriamente dito.

O envelhecimento é influenciado por condições ambientais, pelo volume dos tonéis e barris e pelo tempo de acondicionamento, além da qualidade inicial do destilado. Deve ser

feito em tonéis ou barris de madeira, que oxigenam e arejam a cachaça. O recipiente não deve ser completamente cheio, de modo a manter uma camada de ar na parte superior (SILVA, 2006).

Com relação às condições ambientais, o envelhecimento deve ser em lugar fresco, a uma temperatura na faixa de 15<sup>o</sup>C a 20<sup>o</sup>C, com umidade relativa na faixa de 70% a 90%, além do arejamento adequado (SILVA, 2006). O salão deve ter pé direito alto, em torno de 5 metros, e paredes espessas para evitar oscilações de temperatura, com pequenas janelas, distribuídas para melhor ventilação (SILVA, 2006). Em ambiente seco, haverá tendência de evaporação da água, resultando em acréscimo do grau alcoólico.

É importante destacar que as diferentes madeiras trabalham de forma diferente na liberação de componentes corantes e aromáticos desejáveis ao envelhecimento do destilado e, portanto, requerem tempos diferentes para atingir os padrões de cor e sabor exigidos pelo produtor. Segundo a legislação brasileira, para que a cachaça seja considerada envelhecida, “pelo menos 50% de seu volume precisa ter permanecido por um período mínimo de 12 meses em recipiente de madeira apropriado, com capacidade máxima de 700 litros“. Dependendo da madeira, um longo período de envelhecimento pode provocar a perda do cheiro característico da cana. O período máximo recomendado para que o líquido não perca suas características organolépticas próprias é de 24 meses (SILVA, 2006). Caso a proposta da cachaça seja outra, como impregnar o cheiro de madeira na bebida ou torná-la amarga, esse período pode ser estendido.

## 5.2 Desenvolvimento da Simulação

As condições mostradas favorecem a utilização de RSSF como ferramenta capaz de realizar o monitoramento das condições ambientais durante a fase de envelhecimento da cachaça. Os sensores colocados diretamente no ambiente físico, permitem coletar medidas detalhadas e informações de forma automática, o que seria muito mais difícil de ser executado de forma manual.

O objetivo do estudo de simulação é realizar o monitoramento da fase de envelhecimento da cachaça, observando as condições ambientais de temperatura e umidade pelo período de 12 meses. Com base no que foi exposto na seção anterior, são retiradas as restrições e os requisitos para montar o cenário de simulação.

### 5.2.1 Requisitos da aplicação

O monitoramento da fase de envelhecimento da cachaça apresenta os seguintes requisitos:

#### **Ambiente Físico**

O monitoramento ocorrerá em um galpão de armazenamento de barris de cachaça. As medidas do galpão são 80 metros de comprimento, 8 metros de largura e 5 metros de altura. Os barris utilizados possuem capacidade de 250 litros. Barris com essa capacidade têm dimensões de 95 cm de altura, 72 cm de diâmetro no meio e 58 cm de diâmetro nas extremidades, segundo [Ferreira \(2005\)](#). Os barris estão dispersos em três fileiras duplas (um em cima do outro), conforme ilustrado na Figura 5.2. De acordo com essa distribuição, é possível armazenar até 800 barris no galpão, o que soma um total de 200.000 litros de bebida.

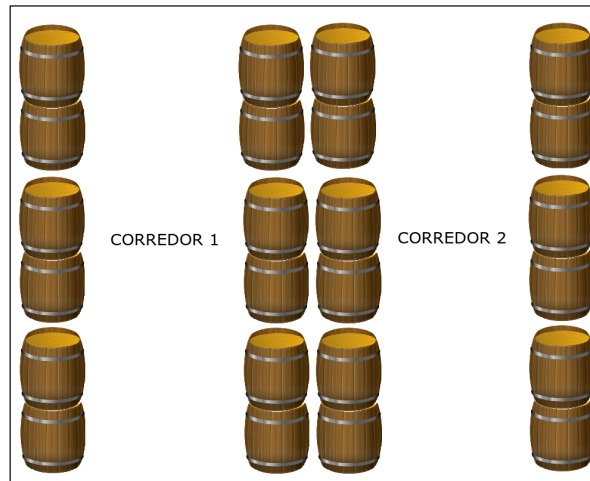


FIGURA 5.2 – Disposição dos barris no galpão

### Monitoramento

Devem ser monitoradas as condições ambientais de temperatura e umidade do galpão. A variação de temperatura permitida oscila entre  $15^{\circ}\text{C}$  e  $20^{\circ}\text{C}$ , e a variação de umidade fica entre 70% e 90%.

### Equipamento Físico

Será utilizado o nó sensor TelosB (POLASTRE; SZEWCZYK; CULLER, 2005), que utiliza o microcontrolador MSP430 e rádio CC2420.

### Aplicação

A aplicação consiste em coletar amostras a cada 15 minutos para verificar se as condições ambientais do galpão estão de acordo com o especificado. Se algum valor coletado estiver fora das especificações, um sinal de alerta deverá ser enviado. Uma vez ao dia um relatório é emitido com a última leitura de temperatura e umidade coletadas.

### Comunicação

O tamanho dos pacotes deve ser de no máximo 100 bytes.

### Manutenção

O monitoramento ocorrerá em ambiente fechado, sem interferências significativas de fatores externos, como chuva e vento. Como o ambiente de monitoramento é de fácil acesso, é possível realizar manutenção nos nós, como troca de bateria, por exemplo.

### **Métricas de avaliação**

*Cobertura:* a rede é considerada confiável enquanto tiver pelo menos 70% de seus nós atuando.

*Consumo de bateria:* será verificado diariamente o nível de bateria que resta nos sensores.

*Tempo de vida:* quantidade maior ou igual a 70% dos nós funcionando.

## **5.2.2 Experimentos Realizados**

Nesta seção apresentam-se os experimentos conduzidos para observar a fase de envelhecimento da cachaça. Foram realizados experimentos no Castalia e TOSSIM utilizando os mesmos parâmetros. Algumas simulações foram executadas no Castalia para que se pudesse obter os resultados adequados. A métrica de Cobertura por exemplo, precisou de 90 sensores para que fosse alcançada. Após ter os resultados esperados, o MD ficou validado e a simulação pôde ser executada também no TOSSIM.

A Tabela 5.1 mostra os parâmetros e os valores constantes no MD para configuração da simulação da aplicação proposta. São mostrados os parâmetros mais relevantes para simplificar o entendimento.

### **Topologia**

Os sensores estão dispostos em forma de *grid*, colocados um pouco acima dos barris,

espaçados um do outro a uma distância aproximada de 2,5 metros na mesma linha, como visualizado na Figura 5.3. De acordo com essa distribuição e dimensões do galpão, são necessários 90 sensores na aplicação.

A aplicação conta com 1 nó *sink*, que fica localizado no centro do campo de sensoria-  
mento.

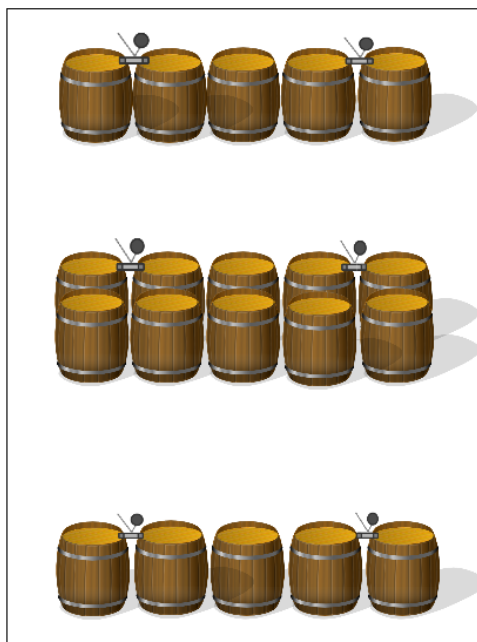


FIGURA 5.3 – Distribuição dos sensores

### Comunicação

Existem vários modelos de canal de transmissão sem fio disponíveis no Castalia. Foi escolhido o modelo realístico por não haver muitas interferências externas. No TOSSIM, é necessário configurar as qualidades das ligações individuais entre os nós, representadas pelos parâmetros *gain* e *noise* explicados na Seção 4.2. Estes parâmetros possuem equivalentes no Castalia.

O tamanho dos pacotes é um parâmetro importante no Castalia, sendo configurado no código fonte do TOSSIM.

### Equipamento Físico

Apesar de ser um requisito a utilização do TelosB, o TOSSIM suporta apenas a plataforma Micaz. Então foi utilizado o rádio CC2420 da plataforma Micaz no Castalia para produzir resultados comparáveis. Foi prevista a troca de bateria dos sensores, caso seja necessário.

TABELA 5.1 – Parâmetros e seus valores para a definição do cenário de simulação

Parâmetro	Valor
Tempo de Simulação	12 meses
Número de Sensores	90 sendo 1 sink
Dimensões	8x80x0
Topologia	Grid
xGridSize	30
yGridSize	4
Número de processos físicos	2
Resultados	Cobertura e GastoEnergia
Processo Físico 1	Temperatura
Processo Físico 2	Umidade
Bateria Inicial	36288 (em Joule)
Equipamento	Micaz
Frequência de Sensoriamento	15 minutos
Rádio	CC2420
Canal de Transmissão	Realístico
Protocolo MAC	TMAC
Protocolo de Roteamento	SimpleTree
MAC-dutyCycle	0.01
Nome Aplicação	WarehouseMonitoring
MaxAppPacketSize	40 (bytes)
PacketHeaderOverhead	8 (bytes)
ConstantDataPayload	8 (bytes)
Resource	2AAbatteries
MAC-listenInterval	1000
MaxNumberOfParents	1
FileGain	topo.txt
FileNoise	meyer-heavy.txt
Attenuation-exp-a	1
OnlyStaticNodes	true

Muitos parâmetros disponíveis no Castalia aparecem como *default* no TOSSIM. Procurou-se modificar os valores dos parâmetros no Castalia para não precisar manusear o código fonte do TOSSIM.

Alguns parâmetros são suportados apenas por um dos simuladores, como por exemplo Resource (especifica os dados da bateria), Nome Aplicação (chama o módulo que implementa a aplicação) e OnlyStaticNodes (especifica que não há movimento dos nós), que são

utilizados somente pelo Castalia. Em contrapartida, FileGain e FileNoise são exclusivos do TOSSIM.

### 5.3 Resultados Obtidos e Análise

Os parâmetros mostrados na Tabela 5.1 foram inseridos no MD. Em seguida, foi aplicada a folha de estilos do Castalia no MD para que o arquivo de configuração do Castalia fosse gerado automaticamente. Aí então a simulação no Castalia foi executada. O próximo passo foi aplicar a folha de estilos do TOSSIM no MD para gerar o arquivo de configuração do TOSSIM e executou-se a simulação no TOSSIM.

A simulação foi executada por um período de 12 meses no Castalia e TOSSIM. Os resultados podem ser visualizados nas Figuras 5.4 e 5.5, que mostram as saídas do Castalia e TOSSIM, respectivamente. A figura 5.6 mostra a comparação dos dois simuladores.

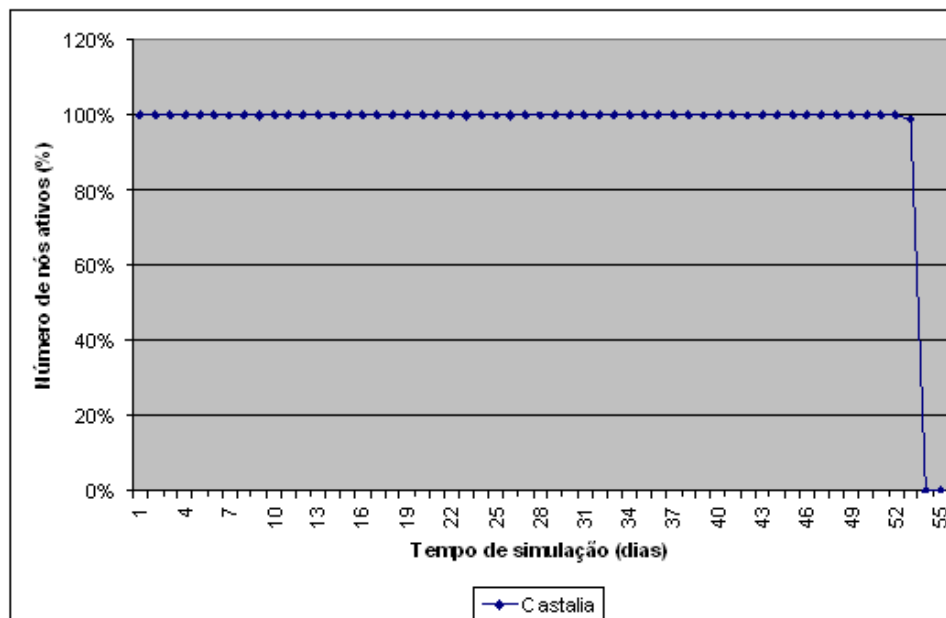


FIGURA 5.4 – Tempo de vida da rede na simulação no Castalia

Verifica-se que o tempo de vida da rede é maior no TOSSIM que chega a 240 dias.



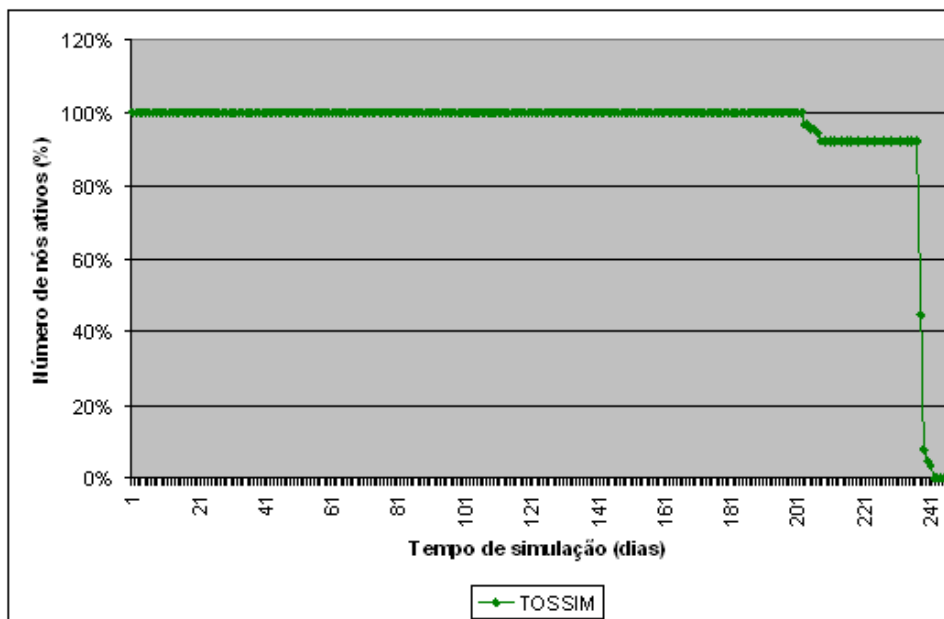


FIGURA 5.5 – Tempo de vida da rede na simulação no TOSSIM

No Castalia, esse tempo é de 53 dias. Uma das causas é o protocolo MAC utilizado no TOSSIM, que tem maior eficiência na economia de energia do que o protocolo TMAC utilizado no Castalia. A simulação provê *reboot* do nó conforme ocorre o término de sua bateria, representando a troca de bateria.

O propósito da análise do estudo de caso é a utilização do MD na fase de configuração da simulação, como modelo extensível de dados de simulação. Verifica-se que foi possível utilizar de maneira satisfatória essa abordagem, pois os modelos gerados utilizaram-se de dados que são comuns aos dois simuladores ou próprios de cada simulador, e os resultados de simulação foram coletados e mostram-se coerentes. O modelo permitiu que os simuladores gerassem resultados consistentes de uma maneira prática, ou seja, os parâmetros da aplicação foram utilizados e as análises puderam ser efetuadas. Com a configuração de um único arquivo, chegou-se a execução da simulação no Castalia e no TOSSIM.

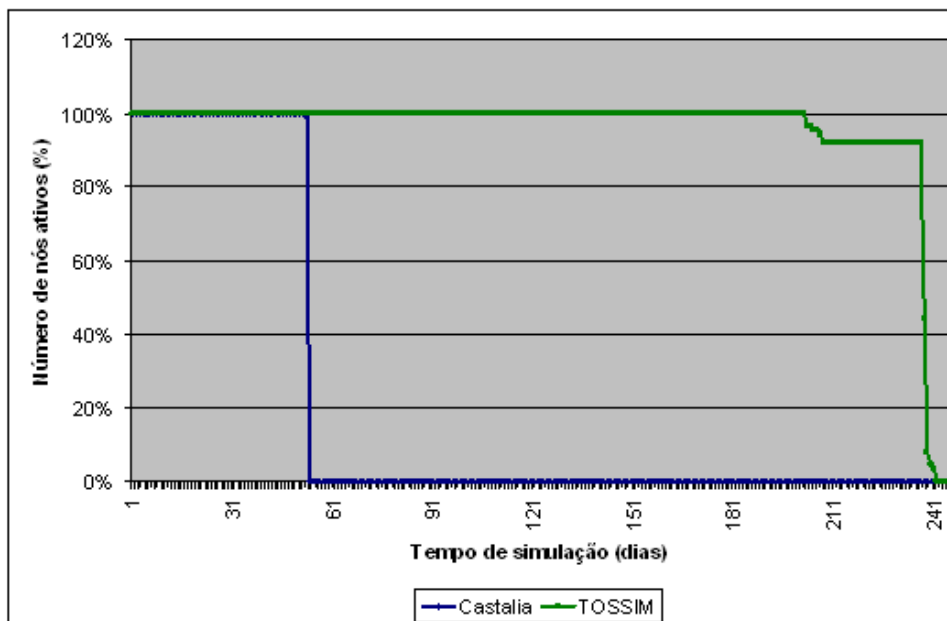


FIGURA 5.6 – Tempo de vida da rede Castalia x TOSSIM

## 5.4 Sumário

Neste capítulo foi avaliada a usabilidade do MD em uma aplicação ilustrativa que fez uso do referido modelo. Foram executadas simulações no Castalia e no TOSSIM com os parâmetros comuns de simulação, especificados no MD, quando requeridos, ou próprios, quando eram particulares de cada modelo.

O MD foi construído e posteriormente transformado no arquivo de configuração de simulação do Castalia e do TOSSIM. Os arquivos de configuração foram criados corretamente e as simulações ocorreram sem impedimentos, com resultados satisfatórios.

O modelo apresentado tem ampla utilização, podendo ser facilmente estendido para incorporar outras ferramentas de simulação.

Verificou-se que essa forma de trabalho é consistente e altera automaticamente os parâmetros dos simuladores contemplados no MD. Foi possível a realização de uma modelagem abstrata, facilitando a alteração de dados.

# 6 Conclusões e Comentários Finais

## 6.1 Conclusões

As RSSFs têm características singulares, diferindo das redes convencionais em vários aspectos, como restrições de energia, grande número de nós e freqüente mudança na topologia devido a falha dos nós. Devido a essas características particulares, a simulação é essencial para o desenvolvimento de aplicações e teste de novos protocolos de RSSF. Contudo, os modelos existentes nos simuladores de RSSFs podem não levar a uma demonstração completa de tudo que precisa ser verificado. Por isso, análises mais confiáveis e precisas podem ser obtidas combinando os resultados de diversos simuladores.

Neste trabalho foi definido um modelo de dados para configuração de simulação, denominado MD (Modelo de Dados). O MD permite realizar a modelagem de cenários de simulação, proporcionando uma forma de modelagem abstrata, onde são colocados os parâmetros de configuração da simulação, podendo ser convertido para o arquivo de configuração específico de cada simulador, através de regras de transformação.

A principal vantagem do MD é a viabilização do compartilhamento de modelos de simulação pelos diversos simuladores. Os simuladores utilizam a mesma configuração de simulação. Quando há necessidade de alterar algum parâmetro de simulação, este é

modificado diretamente no MD, o que faz com que a alteração ocorra automaticamente para todos os simuladores que utilizam o parâmetro.

A avaliação do MD foi conduzida por meio de uma aplicação ilustrativa que utilizou o MD para montar seus arquivos de configuração de simulação. A aplicação consistiu em monitorar a fase de envelhecimento da cachaça, observando as severas condições ambientais impostas pela aplicação. O MD foi montado de acordo com os requisitos e dados de entrada exigidos por cada simulador. Até se chegar aos resultados esperados da simulação, o MD foi sofrendo transformações. Em seguida, utilizando as regras de transformação, chegou-se aos arquivos de configuração dos simuladores Castalia e TOSSIM.

A aplicação utilizou o MD e mostra que essa forma de trabalho é consistente e altera automaticamente para outros simuladores o que precisa ser modificado durante a realização da simulação. Foi possível fazer modelagem em dois simuladores utilizando o mesmo modelo de dados.

## 6.2 Limitações

A atual versão do MD apresenta algumas limitações. Permite utilizar exclusivamente simuladores que provêm algum tipo de arquivo de configuração de simulação; caso contrário, se o código fonte do simulador precisa ser manipulado para realizar configuração de simulação, essa abordagem não é possível. Ainda, existem apenas dois simuladores contemplados na versão atual, sendo necessário implementar uma amostra maior para que se possa tirar maior proveito do modelo unificado.

### 6.3 Sugestões de Trabalhos Futuros

Entre as sugestões para trabalhos futuros destacam-se:

- Incluir mais simuladores no MD, pois existe um grande número de simuladores de RSSF disponíveis que estão aptos para fazer parte do MD.
- Criar um *script* para facilitar a leitura de resultados da simulação.
- Desenvolver uma ferramenta visual, onde usuário escolhe o simulador que deseja utilizar e os parâmetros de entrada exclusivos desse simulador são disponibilizados prontamente para que o usuário preencha seus valores.
- Uma outra opção seria a criação de uma ferramenta onde, levando em consideração os requisitos e objetivos de simulação fornecidos pelo usuário, seria definida automaticamente a escolha do(s) simulador(es) que melhor atende as imposições da aplicação.

# Referências Bibliográficas

ABRACH, H.; BHATTI, S.; CARLSON, J.; DAI, H.; ROSE, J.; SHETH, A.; SHUCKER, B.; DENG, J.; HAN, R. Mantis: system support for multimodal networks of in-situ sensors. In: **WSNA '03: Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications**. New York, NY, USA: ACM, 2003. p. 50–59. ISBN 1-58113-764-8.

AGRAWAL, A.; KARSAL, G.; SHI, F. Graph transformations on domain-specific models. 2003.

AKYILDIZ, I.; SU, W.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. A survey on sensor networks. **IEEE Communications magazine**, Citeseer, v. 40, n. 8, p. 102–114, 2002.

ARCINIEGAS, F. **C++ XML**. [S.l.]: Sams, 2001.

ATMEL. **The AVR 8-Bit RISC Microcontroller**. 1997. Disponível em: <http://www.atmel.com/products/AVR/>. Acesso em: 19 jul. 2009.

BAAR, M.; KÖPPE, E.; LIERS, A.; SCHILLER, J. Poster abstract: The scatterweb msb-430 platform for wireless sensor networks. In: CITESEER. **The Contiki Hands-On Workshop March-2007**. [S.l.], 2006.

BAJAJ, L.; TAKAI, M.; AHUJA, R.; TANG, K.; BAGRODIA, R.; GERLA, M. GloMoSim: A scalable network simulation environment. Citeseer, v. 990027, p. 213, 1999.

BAKSHI, A.; PRASANNA, V. K.; LEDECZI, A. Milan: A model based integrated simulation framework for design of embedded systems. In: **LCTES '01: Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers and Tools for Embedded Systems**. New York, NY, USA: ACM, 2001. p. 82–93. ISBN 1-58113-425-8.

BALCI, O. Validation, verification, and testing techniques throughout the life cycle of a simulation study. In: **WSC '94: Proceedings of the 26th Conference on Winter Simulation**. San Diego, CA, USA: Society for Computer Simulation International, 1994. p. 215–220. ISBN 0-7803-2109-X.

BANKS, J. **Handbook of Simulation - Principles, Methodology, Advances, Applications, and Practice**. Atlanta, Georgia: Engineering and Management Press, 1998.

- BOOCOCPK, P. **JAMDA - Java Model Driven Architecture**. 2003. Disponível em: <<http://jamda.sourceforge.net/>>. Acesso em: 07 abr. 2009.
- BOULIS, A. Castalia version 2.1 user's manual. July 2009.
- BURGER, D.; AUSTIN, T. M. The simplescalar tool set, version 2.0. **SIGARCH Comput. Archit. News**, ACM, New York, NY, USA, v. 25, n. 3, p. 13–25, 1997. ISSN 0163-5964.
- CHANG, X. Network simulations with opnet. In: **WSC '99: Proceedings of the 31st Conference on Winter Simulation**. New York, NY, USA: ACM, 1999. p. 307–314. ISBN 0-7803-5780-9.
- CHEN, G.; BRANCH, J.; PFLUG, M.; ZHU, L.; SZYMANSKI, B. Sense: A sensor network simulator. Citeseer, p. 249–267, 2004.
- CHEONG, E.; LEE, E.; ZHAO, Y. Joint modeling and design of wireless networks and sensor node software. Citeseer, v. 17, p. 2006–150, 2006.
- CONSORTIUM, W. W. W. W. **XML - Extensible Markup Language**. 1998. Disponível em: <<http://www.w3.org/XML/>>. Acesso em: 04 abr. 2007.
- CROSSBOW-TECHNOLOGY. **Mica Motes**. 2002. Disponível em: <<http://www.xbow.com/>>. Acesso em: 17 set. 2009.
- CROSSBOW-TECHNOLOGY. **MICAz wireless measurement system**. 2004. Disponível em: <<http://www.xbow.com/>>. Acesso em: 17 set. 2009.
- CURREN, D. A survey of simulation in sensor networks. **project report (CS580), University of Binghamton**, Citeseer, 2005.
- CZARNECKI, K.; HELSEN, S. Classification of model transformation approaches. In: CITESEER. **Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture**. [S.l.], 2003.
- DOWNARD, I. Simulating sensor networks in ns-2. Citeseer, 2004.
- DULMAN, S.; KAYA, O. S.; KOPRINKOV, G. **EYES WSN Simulation Framework**. 2005. Disponível em: <<http://wwes.cs.utwente.nl/ewnsim/>>. Acesso em: 05 mai. 2009.
- DUNKELS, A. **Contiki in the Baltic Sea**. April 2007. Disponível em: <<http://www.sics.se/contiki/projects/contiki-in-the-baltic-sea.html>>. Acesso em: 17 jul. 2009.
- EGEA-LOPEZ, E.; VALES-ALONSO, J.; MARTINEZ-SALA, A.; PAVON-MARINO, P.; GARCÍA-HARO, J. Simulation tools for wireless sensor networks. In: **Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'05)**. [S.l.: s.n.], 2005.
- EKER, J.; JANNECK, J.; LEE, E.; LIU, J.; LIU, X.; LUDVIG, J.; NEUENDORFFER, S.; SACHS, S.; XIONG, Y. Taming heterogeneity - the Ptolemy approach. **Proceedings of the IEEE**, Citeseer, v. 91, n. 1, p. 127–144, 2003.

- ELSON, J.; BIEN, S.; BUSEK, N.; BYCHKOVSKIY, V.; CERPA, A.; GANESAN, D.; GIROD, L.; GREENSTEIN, B.; SCHOELLHAMMER, T.; STATHOPOULOS, T. *et al.* Emstar: An environment for developing wireless embedded systems software. Citeseer, v. 9, 2003.
- FENG, T. H. Model Transformation with Hierarchical Discrete-Event Control. Technical Report No. UCB/EECS-2009-77, 2009.
- FERREIRA, C. **Vinificação**. 2005. Disponível em: <[http://www.aesbuc.pt/twt/ETGI-MyFiles/MeusSites/Enologia/2005/Madeira\\_vt.htm](http://www.aesbuc.pt/twt/ETGI-MyFiles/MeusSites/Enologia/2005/Madeira_vt.htm)>. Acesso em: 22 nov. 2009.
- GAY, D.; LEVIS, P.; BEHREN, R. von; WELSH, M.; BREWER, E.; CULLER, D. The nesc language: A holistic approach to networked embedded systems. In: **PLDI '03: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation**. New York, NY, USA: ACM, 2003. p. 1–11. ISBN 1-58113-662-5.
- GIROD, L.; STATHOPOULOS, T.; RAMANATHAN, N.; ELSON, J.; ESTRIN, D.; OSTERWEIL, E.; SCHOELLHAMMER, T. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In: **SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems**. New York, NY, USA: ACM, 2004. p. 201–213. ISBN 1-58113-879-2.
- HANDZISKI, V.; KOPKE, A.; KARL, H.; WOLISZ, A. A common wireless sensor network architecture. **Proc. 1. GI/ITG Fachgesprach Sensornetze(Technical Report TKN-03-012 of the Telecommunications Networks Group, Technische Universitat Berlin), Technische Universitat Berlin, Berlin, Germany**, Citeseer, p. 10–17, 2003.
- HILL, J.; CULLER, D. Mica: a wireless platform for deeply embedded networks. **Micro, IEEE**, v. 22, n. 6, p. 12–24, Nov/Dec 2002. ISSN 0272-1732.
- HILL, J.; SZEWCZYK, R.; WOO, A.; HOLLAR, S.; CULLER, D.; PISTER, K. System architecture directions for networked sensors. In: **ASPLOS-IX: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems**. New York, NY, USA: ACM, 2000. p. 93–104. ISBN 1-58113-317-0.
- HOESEL, L. van; DULMAN, S.; HAVINGA, P.; KIP, H. Design of a low-power testbed for wireless sensor networks and verification. 2003.
- ILYAS, M.; MAHGOUB, I. **Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems**. [S.l.]: CRC Press, 2005. ISBN 0-8493-1968-4.
- KÖPKE, A.; SWIGULSKI, M.; WESSEL, K.; WILLKOMM, D.; HANEVELD, P. T. K.; PARKER, T. E. V.; VISSER, O. W.; LICHTTE, H. S.; VALENTIN, S. Simulating wireless and mobile networks in omnet++ the mixim vision. In: **Simutools '08: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & workshops**. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. p. 1–8. ISBN 978-963-9799-20-2.



LARA, J. D.; VANGHELUWE, H. ATOM<sup>3</sup>: A Tool for Multi-formalism and Meta-modelling. **Lecture Notes in Computer Science**, Springer, p. 174–188, 2002.

LAW, A. M.; KELTON, W. D. **Simulation Modeling and Analysis**. [S.l.]: Mc Graw Hill, 2000. ISBN 0-07-059292-6.

LEDECZI, A.; MAROTI, M.; BAKAY, A.; KARSAI, G.; GARRETT, J.; THOMASON, C.; NORDSTROM, G.; SPRINKLE, J.; VOLGYESI, P. The generic modeling environment. In: CITESEER. **Workshop on Intelligent Signal Processing, Budapest, Hungary**. [S.l.], 2001. v. 17.

LEVIS, P.; LEE, N.; WELSH, M.; CULLER, D. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In: ACM NEW YORK, NY, USA. **Proceedings of the 1st International Conference on Embedded Networked Sensor Systems**. [S.l.], 2003. p. 126–137.

LOUREIRO, A.; NOGUEIRA, J.; RUIZ, L.; MINI, R. de F.; NAKAMURA, E.; FIGUEIREDO, C. Redes de sensores sem fio. In: **Simpósio Brasileiro de Redes de Computadores**. [S.l.: s.n.], 2003. p. 179–226.

MALLANDA, C.; SURI, A.; KUNCHAKARRA, V.; IYENGAR, S.; KANNAN, R.; DURRESI, A.; SASTRY, S. Simulating wireless sensor networks with OMNeT++. 2005.

MATHWORKS. **MATLAB**. 1984. Disponível em: <<http://www.mathworks.com/products/matlab/>>. Acesso em: 11 ago. 2009.

MATHWORKS, I. T. **MathWorks**. 1994. Disponível em: <<http://www.mathworks.com/>>. Acesso em: 05 out. 2009.

MULDER, J. **PEEROS: PreEmptive Eyes Real-Time Operating System**. Dissertação (Mestrado) — Twente University, 2003.

OLIVIERI, S. **Model-based design for wireless sensor networks**. 2009. Disponível em: <<http://www.processonline.com.au/articles/33416-Model-based-design-for-wireless-sensor-networks>>. Acesso em: 17 mar. 2009.

ÖSTERLIND, F. **A sensor network simulator for the Contiki OS**. Dissertação (Mestrado) — Swedish Institute of Computer Science, 2006.

OSTERLIND, F.; DUNKELS, A.; ERIKSSON, J.; FINNE, N.; VOIGT, T. Cross-level sensor network simulation with cooja. In: **Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)**. [S.l.: s.n.], 2006. p. 8.

PARK, S.; SAVVIDES, A.; SRIVASTAVA, M. Simulating networks of wireless sensors. In: IEEE COMPUTER SOCIETY WASHINGTON, DC, USA. **Proceedings of the 33rd Conference on Winter Simulation**. [S.l.], 2001. p. 1330–1338.

PERRONE, L. F.; NICOL, D. M. A Scalable Simulator for TinyOS Applications. In: **Proceedings of the 34th Conference on Winter Simulation**. [S.l.]: Institute for Security Technology Studies, 2002. v. 1.

- PIDD, M. **Computer Simulation in Management Science - 5th Edition**. [S.l.]: John Wiley and Sons, Ltd, 2004. ISBN 0-470-09230-0.
- POLASTRE, J.; SZEWCZYK, R.; CULLER, D. Telos: Enabling ultra-low power wireless research. In: IEEE PRESS. **Proceedings of the 4th International Symposium on Information Processing in Sensor Networks**. [S.l.], 2005. p. 48.
- POLLEY, J.; BLAZAKIS, D.; MCGEE, J.; RUSK, D.; BARAS, J.; KARIR, M. Atemu: A fine-grained sensor network simulator. In: CITESEER. **IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks**. [S.l.], 2004.
- RATIONAL, I. **XDE - eXtended Development Environment**. 2006. Disponível em: <<http://www-01.ibm.com/software/awdtools/developer/rose/index.html>>. Acesso em: 07 abr. 2009.
- SARGENT, R. G. Validation and verification of simulation models. In: **WSC '04: Proceedings of the 36th Conference on Winter Simulation**. [S.l.]: Winter Simulation Conference, 2004. p. 17–28. ISBN 0-7803-8786-4.
- SEBRAE-MG. **DIAGNÓSTICO DA CACHAÇA DE MINAS GERAIS**. Relatório Técnico SEBRAE, 2001.
- SILVA, J. M. da. **CACHACA O MAIS BRASILEIRO DOS PRAZERES**. São Paulo, SP: ANHEMBI MORUMBI, 2006.
- SKY, T. **Tmotesky Ultra low power IEEE 802.15.4 compliant wireless sensor module**. 2007. Disponível em: <<http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf>>. Acesso em: 17 set. 2009.
- SOBEIH, A.; CHEN, W.-P.; HOU, J. C.; KUNG, L.-C.; LI, N.; LIM, H.; TYAN, H.-Y.; ZHANG, H. J-sim: A simulation environment for wireless sensor networks. In: **ANSS '05: Proceedings of the 38th Annual Symposium on Simulation**. Washington, DC, USA: IEEE Computer Society, 2005. p. 175–187. ISBN 0-7695-2322-6.
- SRIDHARAN, A.; ZUNIGA, M.; KRISHNAMACHARI, B. Integrating environment simulators with network simulators. **USC Dept. Comp. Sci. tech. rep**, Citeseer, p. 04–386, 2004.
- SUNDRESH, S.; KIM, W.; AGHA, G. Sens: A sensor, environment and network simulator. In: **ANSS '04: Proceedings of the 37th Annual Symposium on Simulation**. Washington, DC, USA: IEEE Computer Society, 2004. p. 221. ISBN 0-7695-2110-X.
- SWEDISH-INSTITUTE-COMPUTER-SCIENCE. **The Contiki Operating System**. 2005. Disponível em: <<http://www.sics.se/contiki/>>. Acesso em: 19 ago. 2009.
- SYSTEMC. **The Open SystemC Initiative (OSCI)**. 1999. Disponível em: <<http://www.systemc.org/home/>>. Acesso em: 11 ago. 2009.

TAKAI, M.; BAGRODIA, R.; TANG, K.; GERLA, M. Efficient wireless network simulations with detailed propagation models. **Wireless Networks**, Springer, v. 7, n. 3, p. 297–305, 2001.

TECHNOLOGIES, S. N. **The Qualnet Simulator**. 2006. Disponível em: <<http://www.qualnet.com/products/qualnet/>>. Acesso em: 19 set. 2009.

TITZER, B. L.; LEE, D. K.; PALSBERG, J. Avroca: scalable sensor network simulation with precise timing. In: **IPSN '05: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks**. Piscataway, NJ, USA: IEEE Press, 2005. p. 67. ISBN 0-7803-9202-7.

UNIVERSITYBERKELEY; LBL; USC/ISI; PARC., X. The ns manual. August 2006.

VARGA, A. *et al.* The OMNeT++ discrete event simulation system. In: **Proceedings of the European Simulation Multiconference (ESM'2001)**. [S.l.: s.n.], 2001. p. 319–324.

VARRÓ, D.; VARRÓ, G.; PATARICZA, A. Designing the automatic transformation of visual languages. **Science of Computer Programming**, Elsevier, v. 44, n. 2, p. 205–227, 2002.

VIEIRA, L. F. **Yatos e wisdom: Plataforma de software para redes de sensores**. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Belo Horizonte, Brazil, 2004.

VIEIRA, M. A. M. **Bean: A computer platform for wireless sensor network**. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, 2004.

VITORINO, B.; ALMEIDA, V. de; VIEIRA, L.; VIEIRA, M.; FERNANDES, A.; JR, D. da S. WISDOM: Desenvolvimento Multiplataforma e Visual para Redes de Sensores sem Fio. 2004.

W3C. **XSLT - XSL Transformations**. 1998. Disponível em: <<http://www.w3.org/XML/>>. Acesso em: 11 nov. 2008.

WERNER-ALLEN, G.; LORINCZ, K.; WELSH, M.; MARCILLO, O.; JOHNSON, J.; RUIZ, M.; LEES, J. Deploying a wireless sensor network on an active volcano. **IEEE Internet Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 10, n. 2, p. 18–25, 2006. ISSN 1089-7801.

WIKIPEDIA. **Compuware. OptimalJ - model-driven development environment for Java**. 2006. Disponível em: <<http://en.wikipedia.org/wiki/OptimalJ>>. Acesso em: 07 abr. 2009.

WIMMER, M.; STROMMER, M.; KARGL, H.; KRAMLER, G. Towards model transformation generation by-example. In: **HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences**. Washington, DC, USA: IEEE Computer Society, 2007. p. 285b. ISBN 0-7695-2755-8.

ZENG, X.; BAGRODIA, R.; GERLA, M. Glomosim: a library for parallel simulation of large-scale wireless networks. In: **PADS '98: Proceedings of the Twelfth Workshop on Parallel and Distributed Simulation**. Washington, DC, USA: IEEE Computer Society, 1998. p. 154–161. ISBN 0-8186-8457-7.

# Anexo A - XML Schema

```
<?xml version="1.0"encoding="UTF-8"standalone="no"?>

<!--W3C Schema generated by XMLSpy v2009 sp1 (http://www.altova.com)-->

<!--Please add namespace attributes, a targetNamespace attribute and import elements according to your requirements-->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"elementFormDefault="qualified">

<xs:import namespace="http://www.w3.org/XML/1998/namespace"/>

<xs:complexType name="Network">

<xs:sequence>

<xs:element ref="Simtime"/>

<xs:element ref="Topology"/>

<xs:element ref="Results"maxOccurs="unbounded"/>

<xs:element ref="Node"maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="id"use="required"type="xs:NMTOKEN"/>

</xs:complexType>

<xs:element name="Network"type="Network"/>

<xs:complexType name="Simtime">

<xs:choice>

<xs:element ref="Time"/>

<xs:element ref="Event"/>

</xs:choice>

<xs:attribute name="unit"fixed="segundos"type="xs:NMTOKEN"/>

<xs:attribute name="initialdelay"type="xs:NMTOKEN"/>

</xs:complexType>

<xs:element name="Simtime"type="Simtime"/>

<xs:complexType name="Time">

<xs:attribute name="unit"fixed="segundos"type="xs:NMTOKEN"/>
```

```
<xs:attribute name="ti" use="required" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Time" type="Time"/>
<xs:complexType name="Event">
<xs:attribute name="ev" use="required" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Event" type="Event"/>
<xs:complexType name="Topology">
<xs:sequence>
<xs:element ref="Type"/>
</xs:sequence>
<xs:attribute name="unit" fixed="metros" type="xs:NMTOKEN"/>
<xs:attribute name="EnvironmentDimensionX" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="EnvironmentDimensionY" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="EnvironmentDimensionZ" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="NumberOfNodes" use="required" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Topology" type="Topology"/>
<xs:complexType name="Type">
<xs:attribute name="name" default="Grid">
<xs:simpleType>
<xs:restriction base="xs:NMTOKEN">
<xs:enumeration value="Grid"/>
<xs:enumeration value="Other"/>
<xs:enumeration value="Uniform"/>
<xs:enumeration value="Random"/>
<xs:enumeration value="UniformRandom"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
<xs:element name="Type" type="Type"/>
<xs:complexType name="Results">
<xs:attribute name="Preferred" default="Perdapacotes">
<xs:simpleType>
```

```
<xs:restriction base="xs:NMTOKEN">
  <xs:enumeration value="Toleranciafalhas"/>
  <xs:enumeration value="Seguranca"/>
  <xs:enumeration value="QualidadeLink"/>
  <xs:enumeration value="Cobertura"/>
  <xs:enumeration value="Perdapacotes"/>
  <xs:enumeration value="Validacaocodigo"/>
  <xs:enumeration value="Latencia"/>
  <xs:enumeration value="Escalabilidade"/>
  <xs:enumeration value="GastoEnergia"/>
  <xs:enumeration value="Exatidao"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="Outro" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Results" type="Results"/>
<xs:complexType name="Node">
  <xs:sequence>
    <xs:element ref="Cenario" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" default="sensor">
    <xs:simpleType>
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="sensor"/>
        <xs:enumeration value="sink"/>
        <xs:enumeration value="target"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="qtd" use="required" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Node" type="Node"/>
<xs:complexType name="Cenario">
  <xs:sequence>
```

```
<xs:element ref="Agent"maxOccurs="unbounded"/>

<xs:element ref="SensorDevice"/>

<xs:element ref="Application"/>

<xs:element ref="Communication"/>

<xs:element ref="Moviment"minOccurs="0"/>

<xs:element ref="Resource"/>

<xs:element ref="Localizacao"minOccurs="0"/>

<xs:element ref="Requirements"minOccurs="0"/>

<xs:element ref="Run"minOccurs="0"maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

<xs:element name="Cenario"type="Cenario"/>

<xs:complexType name="Agent">

<xs:attribute name="qtd"use="required"type="xs:NMTOKEN"/>

<xs:attribute name="FreqEstimulos"type="xs:NMTOKEN"/>

<xs:attribute name="file"type="xs:anySimpleType"/>

<xs:attribute name="nome"type="xs:anySimpleType"/>

<xs:attribute name="padraomov"type="xs:anySimpleType"/>

<xs:attribute name="alcancefenomeno"type="xs:NMTOKEN"/>

<xs:attribute name="PrintDebug"type="xs:anySimpleType"/>

</xs:complexType>

<xs:element name="Agent"type="Agent"/>

<xs:complexType name="SensorDevice">

<xs:sequence>

<xs:element ref="Grandeza"maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="file"type="xs:anySimpleType"/>

<xs:attribute name="FreqSensoriamento"type="xs:NMTOKEN"/>

<xs:attribute name="ConsumoEnergia"type="xs:NMTOKEN"/>

<xs:attribute name="PrintDebug"type="xs:anySimpleType"/>

<xs:attribute name="tipo"default="TelosB">

<xs:simpleType>

<xs:restriction base="xs:NMTOKEN">

<xs:enumeration value="Mica"/>

<xs:enumeration value="TelosB"/>
```

```
<xs:enumeration value="MSB"/>
<xs:enumeration value="TmoteSky"/>
<xs:enumeration value="Mica2"/>
<xs:enumeration value="Outro"/>
<xs:enumeration value="Micaz"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
<xs:element name="SensorDevice" type="SensorDevice"/>
<xs:complexType name="Grandeza">
<xs:attribute name="preferred" default="Temperature">
<xs:simpleType>
<xs:restriction base="xs:NMTOKEN">
<xs:enumeration value="Pressure"/>
<xs:enumeration value="Light"/>
<xs:enumeration value="Other"/>
<xs:enumeration value="Sound"/>
<xs:enumeration value="Moviment"/>
<xs:enumeration value="Humidity"/>
<xs:enumeration value="Temperature"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
<xs:element name="Grandeza" type="Grandeza"/>
<xs:complexType name="Application">
<xs:sequence>
<xs:element ref="Update" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="Reprogram" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="nome" type="xs:anySimpleType"/>
<xs:attribute name="Fidelity" type="xs:NMTOKEN"/>
<xs:attribute name="QtbytesPacote" type="xs:NMTOKEN"/>
<xs:attribute name="QtbytesHeader" type="xs:NMTOKEN"/>
```



```
<xs:attribute name="QtbytesData" type="xs:NMTOKEN"/>
<xs:attribute name="PrintDebug" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Application" type="Application"/>
<xs:complexType name="Update">
<xs:sequence>
<xs:element ref="Evento"/>
</xs:sequence>
<xs:attribute name="Freq" use="required" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Update" type="Update"/>
<xs:complexType name="Evento">
<xs:attribute name="Desc" use="required" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Evento" type="Evento"/>
<xs:complexType name="Reprogram">
<xs:attribute name="Freq" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="Msg" use="required" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Reprogram" type="Reprogram"/>
<xs:complexType name="Communication">
<xs:sequence>
<xs:element ref="WChannel"/>
</xs:sequence>
<xs:attribute name="Roteamento" type="xs:anySimpleType"/>
<xs:attribute name="MAC" type="xs:anySimpleType"/>
<xs:attribute name="PrintDebugMAC" type="xs:anySimpleType"/>
<xs:attribute name="PrintDebugRout" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Communication" type="Communication"/>
<xs:complexType name="WChannel">
<xs:attribute name="file" type="xs:anySimpleType"/>
<xs:attribute name="ParamCollision" type="xs:anySimpleType"/>
<xs:attribute name="ParamInterference" type="xs:anySimpleType"/>
<xs:attribute name="ParamNoise" type="xs:anySimpleType"/>
```

```
<xs:attribute name="ParamOutro" type="xs:anySimpleType"/>
<xs:attribute name="PrintDebug" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="WChannel" type="WChannel"/>
<xs:complexType name="Moviment">
<xs:attribute name="file" type="xs:anySimpleType"/>
<xs:attribute name="onlyStaticNodes" type="xs:anySimpleType"/>
<xs:attribute name="updateInterval" type="xs:NMTOKEN"/>
<xs:attribute name="speed" type="xs:NMTOKEN"/>
<xs:attribute name="initialLocation" type="xs:anySimpleType"/>
<xs:attribute name="pattern" type="xs:anySimpleType"/>
<xs:attribute name="PrintDebug" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Moviment" type="Moviment"/>
<xs:complexType name="Resource">
<xs:sequence>
<xs:element ref="Antenna" minOccurs="0"/>
<xs:element ref="Radio" minOccurs="0"/>
<xs:element ref="Bateria" minOccurs="0"/>
<xs:element ref="LEDS" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
<xs:element name="Resource" type="Resource"/>
<xs:complexType name="Antenna">
<xs:attribute name="marca" use="required" type="xs:anySimpleType"/>
<xs:attribute name="alcance" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Antenna" type="Antenna"/>
<xs:complexType name="Radio">
<xs:attribute name="file" type="xs:anySimpleType"/>
<xs:attribute name="marca" type="xs:anySimpleType"/>
<xs:attribute name="alcance" type="xs:NMTOKEN"/>
<xs:attribute name="tx" type="xs:NMTOKEN"/>
<xs:attribute name="rx" type="xs:NMTOKEN"/>
<xs:attribute name="energiatx" type="xs:NMTOKEN"/>
```

```
<xs:attribute name="energiarx" type="xs:NMTOKEN"/>
<xs:attribute name="PrintDebug" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Radio" type="Radio"/>
<xs:complexType name="Bateria">
<xs:attribute name="file" type="xs:anySimpleType"/>
<xs:attribute name="marca" type="xs:anySimpleType"/>
<xs:attribute name="energiainicial" type="xs:NMTOKEN"/>
<xs:attribute name="PrintDebug" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Bateria" type="Bateria"/>
<xs:complexType name="LEDS">
<xs:attribute name="value" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="LEDS" type="LEDS"/>
<xs:complexType name="Localizacao">
<xs:attribute name="locX" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="locY" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="locZ" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="granularidade" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Localizacao" type="Localizacao"/>
<xs:complexType name="Requirements">
<xs:sequence>
<xs:element ref="Tempovida" minOccurs="0"/>
<xs:element ref="Sincronizacao" minOccurs="0"/>
<xs:element ref="Custo" minOccurs="0"/>
<xs:element ref="Recarga" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="AgregDados" minOccurs="0"/>
<xs:element ref="Seguranca" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
<xs:element name="Requirements" type="Requirements"/>
<xs:complexType name="Tempovida">
<xs:attribute name="qtdmin" type="xs:anySimpleType"/>
```

```
<xs:attribute name="outro" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Tempovida" type="Tempovida"/>
<xs:complexType name="Sincronizacao">
  <xs:attribute name="granularidade" use="required" type="xs:NMTOKEN"/>
  <xs:attribute name="freq" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Sincronizacao" type="Sincronizacao"/>
<xs:complexType name="Custo">
  <xs:attribute name="valor" use="required" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Custo" type="Custo"/>
<xs:complexType name="Recarga">
  <xs:attribute name="qtd" use="required" type="xs:NMTOKEN"/>
  <xs:attribute name="nos" use="required" type="xs:NMTOKEN"/>
  <xs:attribute name="freq" use="required" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Recarga" type="Recarga"/>
<xs:complexType name="AgregDados">
  <xs:attribute name="protocolo" use="required" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="AgregDados" type="AgregDados"/>
<xs:complexType name="Seguranca">
  <xs:attribute name="protocolo" use="required" type="xs:NMTOKEN"/>
  <xs:attribute name="outro" type="xs:NMTOKEN"/>
</xs:complexType>
<xs:element name="Seguranca" type="Seguranca"/>
<xs:complexType name="Run">
  <xs:sequence>
    <xs:element ref="Random" minOccurs="0"/>
    <xs:element ref="Outros" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="Run" type="Run"/>
<xs:complexType name="Random">
```

```
<xs:sequence>
  <xs:element ref="seeds" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="distribution" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Random" type="Random"/>
<xs:complexType name="seeds">
  <xs:attribute name="seed" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="seeds" type="seeds"/>
<xs:complexType name="Outros">
  <xs:attribute name="parametro" use="required" type="xs:anySimpleType"/>
</xs:complexType>
<xs:element name="Outros" type="Outros"/>
</xs:schema>
```

## FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO <p style="text-align: center;">DM</p>	2. DATA <p style="text-align: center;">25 de janeiro de 2010</p>	3. DOCUMENTO No. <p style="text-align: center;">CTA/ITA/DM-121/2009</p>	4. No. DE PÁGINAS <p style="text-align: center;">93</p>
5. TÍTULO E SUBTÍTULO: Um Modelo de Dados para Configuração da Simulação no Desenvolvimento de Aplicações de Redes de Sensores Sem Fio			
6. AUTOR(ES): <b>Luciana Brasil Rebelo dos Santos</b>			
7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES): Instituto Tecnológico de Aeronáutica. Divisão de Ciência da Computação – ITA/IEC.			
8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR: Simulação Discreta; Redes de Sensores Sem Fio; Modelo de Dados Extensível.			
9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: Simulação de eventos discretos; Sensores; Comunicação sem fio; Redes de comunicação; Modelos de dados; Engenharia eletrônica; Computação.			
10. APRESENTAÇÃO: <span style="float: right;"><input checked="" type="checkbox"/> Nacional    <input type="checkbox"/> Internacional</span> ITA, São José dos Campos. Curso de Mestrado. Programa de Pós-Graduação em Engenharia Eletrônica e Computação. Área de Informática. Orientadores: Celso Massaki Hirata e Vakulathil Abdurahiman (in memorian). Defesa em 18/12/2009. Publicada em 2009.			
11. RESUMO: <p>A simulação possibilita a realização de uma análise detalhada de Redes de Sensores Sem Fio (RSSFs) de forma rápida e econômica, tornando-se uma importante opção para estudo destas redes. Apesar do grande número de simuladores de RSSFs disponíveis, cada um deles foi projetado para atender necessidades específicas de seus criadores. Por causa dos aspectos singulares das RSSFs, os modelos existentes nos simuladores podem não ser precisos e completos para todas as áreas de simulação de RSSFs, como por exemplo, análise de escalabilidade, modelo de mobilidade, ambiente físico, consumo de energia e condições de tempo. Adicionalmente, a integração entre esses simuladores geralmente é pequena ou inexistente. Nesse contexto, este trabalho apresenta uma forma de modelagem abstrata de cenários de simulação de RSSFs, permitindo que um modelo de dados de configuração de simulação seja compartilhado por mais de um simulador, viabilizando a integração entre eles. A avaliação do modelo de dados proposto foi feita através do desenvolvimento de uma aplicação ilustrativa, onde o modelo de dados foi compartilhado por dois simuladores, que utilizaram um modelo de cenários e produziram adequadamente seus respectivos resultados de simulação.</p>			
12. GRAU DE SIGILO: <b>(X) OSTENSIVO</b> <input type="checkbox"/> RESERVADO <input type="checkbox"/> CONFIDENCIAL <input type="checkbox"/> SECRETO			