Pós-Graduação em Ciência da Computação

Júlio Rodrigues de Mendonça Neto

MODELAGEM E ANÁLISE DE DESEMPENHO E CONSUMO DE ENERGIA EM APLICAÇÕES MÓVEIS

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2015

Júlio Rodrigues de Mendonça Neto

MODELAGEM E ANÁLISE DE DESEMPENHO E CONSUMO DE ENERGIA EM APLICAÇÕES MÓVEIS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Prof. Dr. Ricardo Massa Ferreira Lima*
Co-Orientador: *Prof. Dr. Ermeson Carneiro de Andrade*

RECIFE
2015

Catalogação na fonte
Bibliotecário Jefferson Luiz Alves Nazareno CRB4-1758

M539m Mendonça Neto, Júlio Rodrigues de.
Modelagem e análise de desempenho e consumo de energia em aplicações móveis/ Júlio Rodrigues de Mendonça Neto. – RECIFE: O Autor, 2015.
92 f.: fig.,tab.

Orientador Prof. Dr. Ricardo Massa Ferreira Lima
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIN, Ciência da Computação, 2015.
Inclui Referências.

1. Engenharia de Software. 2. Avaliação de Desempenho (Computadores). 3. Teoria da Computação. I. Lima, Ricardo Massa Ferreira (Orientador). II. Título.

005.1 CDD (22. ed.) UFPE-MEI 2015-122

Dissertação de mestrado apresentada por **Júlio Rodrigues de Mendonça Neto** ao programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **Modelagem e análise de desempenho e consumo de energia em aplicações móveis**, orientada pelo **Prof. Dr. Ricardo Massa Ferreira Lima** e aprovada pela banca examinadora formada pelos professores:

Prof. Ricardo Massa Ferreira Lima
Centro de Informática/UFPE

Prof. Abel Guilhermino da Silva Filho
Centro de Informática/UFPE

Prof. Gustavo Rau de Almeida Callou
Departamento de Estatística e Informática/UFRPE

RECIFE
2015

À meus pais e irmãos.

Agradecimentos

A Deus, por todas as graças concedidas.

Ao meu orientador, Prof. Ricardo Massa, por aceitar o desafio de me orientar, pelo ótimo acolhimento, paciência, amizade e principalmente pela confiança em mim e no meu trabalho. Agradeço também ao meu co-orientador, Prof. Ermeson Andrade, pelo apoio nessa caminhada, por sempre estar disponível para ajudar, aconselhar, pela paciência e amizade.

Aos meus pais, Diana e Júlio, pelo apoio, incentivo, por acreditarem em mim, por me dar coragem, força e carinho para seguir meu caminho. Mesmo diante de tantas dificuldades, vocês sempre fazem de tudo para apoiar seus filhos e nos ajudar a vencer nossas batalhas. Sem vocês eu nada seria. Não poderia ter melhores pais. Eu amo vocês.

A minha toda minha família, por todo apoio, encorajamento, suporte e amizade. De forma especial, aos meus irmãos, Marco e Felipe, e a minha prima, Izabel Noberto, pelo suporte e alegrias compartilhadas nesse tempo morando juntos. Aos meus tios, Fátima e Milton, e meus primos Arthur e Gabriel, que me acolheram da melhor forma possível em sua casa, em meus primeiros anos morando em Recife. Vocês, e todos os outros de nossa família não citados aqui (são muitos), dão mais sentido a vida.

Aos amigos que fiz no Cin-UFPE e no MoDCS, em especial a Airton Pereira, Carlos Magno, Clara Bezerra, Danilo Oliveira, Eliomar Campos, Edson Alves, Érico Augusto, Érica Sousa, Leandro Marques, Igor Oliveira, Jamilson Dantas, Rosângela Melo e Verônica Conceição por todo o convívio, amizade, discussões, apoio, ajuda e momentos de descontração.

Aos amigos do NUTES-UFPE, em especial a Agay Borges, Aloísio Soares, Beatriz Melo, Danilo Soares, Fernando Sales, Pedro Melo, Rômulo Ferreira e Yago Nobre pelas discussões e momentos de descontração. E também a todos os amigos de longa data, em especial a Alan Cesario, Dreyton Henrique, Eberth Lopes, Jair Galvão, Luiz Valdo, Paulo Marcelo, Rodrigo Ramos pelo incentivo para iniciar o mestrado, pelo apoio durante o curso e pelos bons momentos de descontração.

Ao Cin-UFPE e a FACEPE, pelo apoio e recursos recebidos.

(...) Se fosse fácil achar o caminho das pedras, tantas pedras no caminho não seria ruim.

—HUMBERTO GESSINGER

Resumo

É notável a propagação de dispositivos móveis inteligentes em todo o mundo. Em 2016, espera-se que o número de *smartphones* supere os 2 bilhões. A entrada desses aparelhos no mercado têm mudado o estilo de vida das pessoas, oferecendo soluções com mais facilidade e praticidade, como por exemplo, a realização de transações bancárias. A variedade e facilidade de acesso aos serviços oferecidos pelos mesmos têm ajudado nesta mudança. Além disso, o número de aplicativos nos *marketplaces* para satisfazer as mais diferentes necessidades dos usuários tem aumentado bastante. Por estes motivos, as aplicações móveis têm ganhado cada vez mais destaque na indústria de Tecnologia da Informação e Comunicação (TIC). Contudo, apesar da evolução da computação móvel na última década, a tecnologia das baterias desses dispositivos não evoluíram na mesma velocidade. Sendo, portanto, o curto tempo de vida das baterias uma das maiores preocupações entre fabricantes de *hardware* e *software* para dispositivos móveis. É comum o uso de modelos analíticos para analisar o comportamento dos sistemas. Modelos formais como cadeias de Markov e redes de Petri são bastante utilizadas com esse propósito. Estes tipos de modelos tem fundamentação matemática sólida, e portanto, são eficientes para realização de análises quantitativas e verificação de propriedades dos sistemas representados. Entretanto, sua construção requer conhecimento especializado por parte dos projetistas de sistemas. Por outro lado, modelos semiformais, como *System Modelling Language* (SysML) e *Unified Model Language* (UML), possuem uma notação mais flexível e são bastante difundidos no mercado. Contudo, esses modelos semiformais, por não possuírem uma fundamentação formal rígida, não oferecem suporte para o desenvolvimento de técnicas para análises numéricas e verificações de propriedades. Dessa forma, torna-se interessante a combinação do uso de modelos formais e modelos semiformais. Nesse contexto, este trabalho propõe uma abordagem para avaliação de métricas de desempenho, consumo de energia e disponibilidade de aplicações móveis utilizando modelos estocásticos. Para facilitar a construção destes modelos, a abordagem proposta adota a criação de regras de mapeamento, para obtenção de Redes de Petri Estocásticas e Determinísticas (DSPNs) a partir de diagramas da SysML. Desta forma, projetistas que possuem pouco conhecimento em modelagem estocástica podem realizar análises relacionadas ao desempenho e consumo de energia de aplicações móveis de forma rápida e menos custosa que o desenvolvimento de protótipos, ainda na fase de planejamento do projeto. Por fim, são apresentados estudos de casos que demonstram a aplicabilidade da abordagem.

Palavras-chave: Análise de desempenho. Consumo de energia. Aplicações móveis. Modelos. SysML. Redes de Petri.

Abstract

It is remarkable smart mobile devices spread around the world. In 2016, it is expected that the number of smartphones exceeds 2 billion. These devices entrace on the market have changed the people lifestyle, offering solutions with more ease and practicality, for example, conducting banking transactions. The variety and easy access to services offered by them have helped this change. Also, the number of applications in the marketplaces to meet the many different users needs has greatly increased. For these reasons, mobile applications have gained more prominence in the Information and Communication Technology (ICT) industry. However, despite the mobile computing progress in the last decade, the battery technology of these devices have not evolved at the same speed. It is, therefore, the short lifetime of batteries a major concern between hardware and software manufacturers to mobile devices. Analytical models are usually used to analyze the systems behavior. Formal models as Markov chains and Petri nets are widely used for this purpose. These types of models have a solid mathematical basis, and therefore, are effective in carrying out quantitative analyzes and verification systems properties. Nevertheless, their construction requires specialized knowledge by the system designers. Moreover, semi-formal models, such as System Modelling Language (SysML) and Unified Model Language (UML) have a more flexible notation and are fairly widespread in the market. Meanwhile, these semi-formal models, for not having a rigid formal basis, do not support techniques for numerical analysis and property checks. Therefore, it becomes interesting to use the combination of formal and semi-formal models. In this context, this work proposes an approach to evaluation performance, power consumption and availability metrics of mobile applications using stochastic models. In order to facilitate construction of such models, the proposed approach adopts the creation of mapping rules for obtaining Stochastic Petri Nets and deterministic (DSPNs) from SysML diagrams. In this way, designers who have little knowledge in stochastic modeling can perform analyzes related to the performance and power consumption in mobile applications quickly and less costly to develop prototypes, still in the project planning phase. Finally, case studies are presented to demonstrate the approach applicability.

Keywords: Performance Analysis. Energy consumption. Mobile applications. Models. SysML. Petri nets.

Lista de Figuras

2.1 Classificação das técnicas de análises de desempenho. 27
2.2 Déficits dos dispositivos móveis. 30
2.3 Arquitetura de uma MCC. 32
2.4 Estrutura da SysML. 33
2.5 Quadro para modelagem dos Diagramas da SysML. 34
2.6 Exemplo de um SysML-IBD. 34
2.7 Exemplo de um SysML-STM. 35
2.8 Tipos de redes de Petri. 37
2.9 Exemplo de uma rede de Petri *Place/Transition*. 38
2.10 Exemplo de Disponibilidade em RBD. 43
2.11 Tipos de agrupamentos de componentes em um RBD. 43

3.1 Abordagem adotada neste trabalho 46
3.2 Exemplo de mapeamento de um bloco do SysML-IBD em uma DSPN. 47
3.3 Exemplo de mapeamento de blocos com redundância. 48
3.4 Regras de mapeamento dos elementos do SysML-STM em DSPNs. 49
3.5 Exemplo de mapeamento de um SysML-STM. 51
3.6 Exemplo de um SysML-STM com um estado composto 53
3.7 Mapeamento dos SysML-STM da Figura 3.6. 54

4.1 Cenário base adotado para estudo 56
4.2 Exemplo de código em Java utilizado para obter tempo de execução de alguma funcionalidade. 56
4.3 *PowerTutor* em execução em dispositivo Android. 57
4.4 Modelo RBD representando o *access point* e o sinal do *access point*. 61
4.5 SysML-IBD correspondente a Figura 4.1. 61
4.6 SysML-STM correspondente ao *access point*. 62
4.7 SysML-STMs representando o comportamento do *smartphone* e da aplicação móvel. 63
4.8 Mapeamento do bloco MCC. 63
4.9 DSPN correspondente ao **stm *access point*** mostrado na Figura 4.6. 64
4.10 DSPNs correspondentes aos SysML-STMs da Figura 4.7. 65

5.1 Resultados de tempo médio de execução dos experimentos e da simulação da DSPN 68
5.2 Cenário ilustrando *upload* de imagens via diferentes interfaces de rede. 69

5.3 SysML-IBD correspondente a Figura 5.2. 70
5.4 SysML-STM correspondente ao comportamento do *access point*. 71
5.5 SysML-STMs representando o comportamento do *smartphone*, e o detalhamento do estado composto representando o comportamento da aplicação móvel 71
5.6 DSPNs obtidas a partir do mapeamento dos diagramas SysML criados. 72
5.7 Cenário adotado para estudo, contendo uma arquitetura de *cloudlet*. 75
5.8 SysML-IBD representando o cenário da Figura 5.7. 75
5.9 DSPNs obtidas a partir do mapeamento dos diagramas SysML criados. 76
5.10 Comparação da arquitetura base com compressão de imagens e sem compressão de imagens. 79
5.11 Comparação da arquitetura com *upload* via múltiplas interfaces de rede, com compressão de imagens e sem compressão de imagens. 80
5.12 Comparação da arquitetura de *cloudlet* utilizando *upload* com compressão de imagens e sem compressão de imagens. 80

Lista de Tabelas

1.1 Comparação de características entre os trabalhos relacionados. 24

3.1 Funções de guarda utilizadas na DSPN da Figura 3.7 (b). 54

4.1 Número mínimo de experimentos necessários para captura de imagens. 59
4.2 Número mínimo de experimentos necessários para compressão de imagens. . . 59
4.3 Número mínimo de experimentos necessários para *upload* de imagens via WLAN. 59
4.4 Valores obtidos com os experimentos para arquivos de 1MB 60
4.5 Parâmetros utilizados nos modelos criados. 60
4.6 Parâmetros agrupados. 61
4.7 Funções do TimeNET utilizadas para obtenção das métricas. 65
4.8 Função de guarda utilizada na DSPN da Figura 4.10 (b). 66

5.1 Funções do TimeNET utilizadas para obtenção das métricas. 69
5.2 Funções de guarda da DSPN mostrada na Figura 5.6 (a). 73
5.3 Parâmetros de tempo médio de execução e consumo médio de energia. 73
5.4 Equações do TimeNET utilizadas para obtenção das métricas do estudo de caso. 74
5.5 Funções de guarda da DSPN mostrada na Figura 5.9 (a). 77
5.6 Parâmetros de tempo médio de execução e consumo médio de energia do estudo de caso. 77
5.7 Equações usadas no TimeNET para obtenção das métricas do estudo de caso. . 78
5.8 Parâmetros para *upload* de imagem (.jpg) de 1MB. 79
5.9 Métricas estimadas pelos modelos nas arquiteturas utilizadas. 81

Lista de Acrônimos

SO Sistema Operacional 19

API *Application Programming Interface* 29

SDK *Software Development Kit* 29

DSPN Rede de Petri Estocástica e Determinística 18

SPN Rede de Petri Estocástica 18

ETPN Rede de Petri Temporizada com restrição de Energia 21

SysML *System Modelling Language* 18

SysML-AD Diagrama de Atividades da SysML 45

SysML-STM Diagrama de Estados da SysML 35

SysML-IBD Diagrama de Bloco Interno da SysML 33

RBD Diagrama de Blocos de Confiabilidade 21

MARTE *Profile for Modeling and Analysis of Real-time and Embedded systems* 18

MTTF Tempo Médio até a falha 46

MTTR Tempo Médio de reparo 46

UML *Unified Model Language* 17

VM Máquina Virtual 28

MCC *Mobile Cloud Computing* 17

WLAN Rede de área local sem-fio 22

WiFi Rede sem-fio 21

MB *Megabyte* 58

Mbps *Megabits por segundo* 58

2G Segunda geração de padrões e tecnologias de telefonia móvel 19

3G Terceira geração de padrões e tecnologias de telefonia móvel 19

4G Quarta geração de padrões e tecnologias de telefonia móvel 19

TIC Tecnologia da Informação e Comunicação 16

TI Tecnologia da Informação 17

CPU Unidade de Processamento Central 21

NIST Instituto Nacional de Padrões e Tecnologia dos Estados Unidos 27

SaaS *Software* como Serviço 28
PaaS Plataforma como Serviço 28
IaaS Infraestrutura como Serviço 28
DaaS Dados como Serviço 28
MBaaS *Mobile Backend* como Serviço 28
GRM Modelagem de Recursos Genéricos 36

Sumário

1 Introdução **16**

1.1 Motivação e Justificativa 18
1.2 Trabalhos Relacionados 19
1.3 Objetivos 25
1.4 Organização da Dissertação 25

2 Fundamentação Teórica **26**

2.1 Avaliação de desempenho e consumo de energia 26
2.2 Fundamentos sobre *Mobile Cloud Computing* 27
2.2.1 Computação em nuvem 27
2.2.2 *Mobile Cloud Computing* 29
2.3 SysML 31
2.3.1 SysML-IBD 33
2.3.2 SysML-STM 35
2.4 MARTE 36
2.5 Redes de Petri 36
2.5.1 Propriedades das Redes de Petri 38
2.5.1.1 Propriedade Comportamentais 39
2.5.1.2 Propriedade Estruturais 39
2.5.2 Extensões Temporizadas de Redes de Petri 40
2.5.2.1 Redes de Petri Determinísticas e Estocásticas 41
2.6 Diagrama de blocos de Confiabilidade 42
2.7 Considerações Finais 44

3 Mapeamento dos Diagramas da SysML em DSPNs **45**

3.1 Abordagem adotada 45
3.2 Mapeamento do SysML-IBD 47
3.3 Mapeamento do SysML-STM 48
3.3.1 Mapeamento de Estados Compostos 53
3.4 Considerações Finais 54

4 Modelos e Experimentos **55**

4.1 Visão Geral 55
4.2 Obtenção dos parâmetros de tempo de execução e consumo de energia 56
4.2.1 Experimentos realizados 57
4.3 Modelos gerados 60

4.4 Considerações Finais 66

5 Estudo de Casos **67**
5.1 Validação dos Modelos 67
5.2 Estudo de Caso 1: *Upload* de arquivos com múltiplas interfaces de rede 69
5.2.1 Criação dos diagramas SysML e Mapeamento em DSPNs 70
5.2.2 Análises numéricas 73
5.3 Estudo de Caso 2: *Upload* de arquivos com múltiplas interfaces de rede e *Cloudlet* 74
5.3.1 Criação dos diagramas SysML e Mapeamento em DSPNs 75
5.3.2 Análises numéricas 77
5.4 Estudo de caso 3: *Upload* de arquivos não comprimidos 78
5.5 Comparação entre as arquiteturas utilizadas 81
5.6 Considerações Finais 81

6 Conclusões e Trabalhos Futuros **83**
6.1 Contribuições 84
6.2 Limitações 85
6.3 Trabalhos Futuros 85

Referências **87**

1 Introdução

You can't connect the dots looking forward. You can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future.

—STEVE JOBS

Os dispositivos móveis revolucionaram a área da telecomunicação. Com o surgimento de dispositivos móveis inteligentes (tais como: *smartphones* e *tablets*), nas últimas décadas, um novo marco na evolução das telecomunicações foi fixado. A difusão desses aparelhos tem crescido num ritmo acelerado em todo o mundo e espera-se que o número de usuários de *smartphones* supere os 2 bilhões em 2016 (EMARKETER, 2014). Essa popularização tem mudado o comportamento dos usuários para acessar os serviços que eles necessitam e, consequentemente, da indústria de telecomunicações e Tecnologias da Informação e Comunicação (TICs) para produzir e disponibilizar seus serviços e produtos no mercado. Um relatório da *Pew Research Center* (Pew Research Center, 2015) mostra que cerca de 62% dos americanos utilizam os dispositivos móveis para buscar informações sobre saúde e cerca de 57% utilizam esses dispositivos para realizar ou consultar serviços bancários. A priorização das empresas em disponibilizar seus serviços através das plataformas móveis se torna evidente quando, ainda em 2014, foi observado que o número de acessos a conteúdos *online* através de plataformas *mobile* superou o número de acessos via *desktops* (COMSCORE, 2014).

Alinhado a expansão da computação móvel, a computação em nuvem também tem crescido e sido amplamente reconhecida como a próxima geração de infraestrutura computacional (DINH et al., 2011). A computação em nuvem traz um modelo que fornece conveniência e provê acesso sob demanda, através da rede, a um conjunto de recursos computacionais compartilhados e configuráveis, que podem ser rapidamente disponibilizados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços (BADGER et al., 2012). Dessa maneira, esse modelo provê a entrega de infraestrutura e sistemas de software como serviço através da Internet, fornecendo uma grande quantidade de recursos computacionais, que estão atreladas a

uma alta disponibilidade (ARMBRUST et al., 2010).

O mercado tem notado os benefícios trazidos pela computação em nuvem. Uma pesquisa realizada com empresas de diversos ramos, mostrou que cerca de 82% delas afirmam ter economizado dinheiro migrando suas soluções para a nuvem (SANTACATERINA, 2013). Em 2014, mais de 60% das empresas utilizavam computação em nuvem para operações relacionadas à Tecnologia da Informação (TI) (WALDEN, 2015). O mercado da computação em nuvem cresceu de $46 bilhões de dólares em 2008, para mais de $150 bilhões de dólares em 2014 (WALDEN, 2015). Já em 2016, espera-se que os maiores gastos com TI estejam relacionados à computação em nuvem (SHETTY, 2013).

O crescimento conjunto da computação móvel e da computação em nuvem fez com que em meados de 2007 fosse introduzido o paradigma da *Mobile Cloud Computing* (MCC) (DINH et al., 2011). O principal objetivo da MCC é o provimento de recursos computacionais das nuvens para aumentar o poder das tarefas executadas pelos dispositivos móveis. Este paradigma facilita o uso de o uso do *offloading* (KUMAR et al., 2013), uma técnica para a particionamento das tarefas entre o dispositivo móvel e um ambiente de nuvem. Aliada a computação em nuvem, a MCC vem sendo utilizada, provendo recursos e técnicas, para ajudar na diminuição do consumo de energia dos dispositivos móveis (RAHMAN; GAO; TSAI, 2013).

Nos últimos anos, o consumo de energia dos dispositivos móveis têm se tornado uma grande preocupação entre os fabricantes de *hardwares* e *softwares*. Mesmo com o avanço da tecnologia para os componentes de *hardware* desses dispositivos (processadores, memória, *displays*), o tempo de vida da bateria ainda continua sendo um fator crítico (OLIVEIRA et al., 2013). O desenvolvimento de novas tecnologias para baterias desses dispositivos é mais lento e não consegue acompanhar a velocidade dos outros componentes de *hardware* (SHERR; TIBKEN, 2014). Até mesmo entre os usuários finais, o tempo de vida da bateria é um fator relevante na hora de se comprar um *smartphone* (PHONES, 2013). Além disso, mesmo com o aumento dos recursos computacionais dos aparelhos móveis, as aplicações passaram a exigir cada vez mais deles, agregando cada vez mais funcionalidades e mais complexidade as funcionalidades. Milhares de aplicações são criadas e disponibilizadas todos os dias nas lojas de aplicativos (*marketplaces*). Contudo, nem todas as aplicações são construídas levando-se em conta sua eficiência energética.

Modelos analíticos vêm sendo utilizados para modelagem e análise de sistemas complexos. Cadeias de Markov (TRIVEDI, 1982) e redes de Petri (MURATA, 1989) têm sido bastante utilizadas para este propósito. Esses modelos possuem uma fundamentação matemática sólida e, portanto, são eficientes para realização de análises quantitativas e qualitativas além de verificações de propriedades. Tais modelos podem auxiliar em comparações de alternativas para as soluções e nos ajustes dos sistemas de forma rápida, com baixo custo financeiro e computacional e com um bom nível de confiabilidade. Contudo, a modelagem analítica necessita de conhecimento especializado por parte dos projetistas, o que nem sempre acontece. Por outro lado, a modelagem através de modelos semiformais utilizando *Unified Model Language* (UML)

(OMG, 2004) e *System Modelling Language* (SysML) é bastante difundida no mercado e muito usada durante o processo de desenvolvimento de *softwares*. A modelagem usando linguagens semiformais apresentam algumas vantagens: (i) ajudam a visualizar o sistema como um todo, (ii) permitem modelar a estrutura ou o comportamento de um sistema e (iii) proporcionam um guia para a construção/entendimento de um sistema (ANDRADE, 2009).

Neste contexto, este trabalho propõe uma abordagem para realização de análise de desempenho, consumo de energia e disponibilidade de aplicações móveis, utilizando o mapeamento de modelos semiformais em modelos analíticos. Essa abordagem propõe o mapeamento dos diagramas da SysML (HAUSE, 2006) anotados com o *Profile for Modeling and Analysis of Real-time and Embedded systems* (MARTE) (OMG., 2011) em Redes de Petri Estocásticas e Determinísticas (DSPNs) (MARSAN; CHIOLA, 1987) para obtenção de métricas de desempenho, consumo de energia e disponibilidade. O *profile* MARTE é utilizado em conjunto com os diagramas da SysML na criação dos modelos semiformais. Ele torna possível a utilização de anotações de domínio específico nos diagramas da SysML, auxiliando na definição de propriedades dos sistemas a serem analisados. Já a DSPN é um tipo de rede Petri que possui o tempo como propriedade de transições determinísticas e estocásticas (MARSAN; CHIOLA, 1987). Diferente das Redes de Petri Estocásticas (SPNs) (MARSAN, 1990), esse tipo de rede de Petri possui a transição determinística, na qual o tempo atribuído a transição é fixo, nas transições estocásticas o tempo é exponencialmente distribuído. Redes de Petri são consideradas uma ferramenta poderosa para modelagem de sistemas complexos por oferecer mecanismos capazes de modelar concorrência, filas, paralelismo e sincronização (ZIMMERMANN; BODE; HOMMEL, 1996). Essas redes têm sido bastante utilizadas ao longo do tempo para realização de análises de desempenho de sistemas (SIFAKIS, 1977).

Portanto, a abordagem proposta neste trabalho pode ajudar em um melhor planejamento das aplicações móveis quanto ao desempenho, bem como ao consumo de energia e disponibilidade. Mesmo oferecendo uma construção mais fácil dos modelos, as linguagens semiformais não oferecem métodos para análises numéricas. Sendo assim, a combinação do uso de modelos semiformais e modelos analíticos é interessante. Neste trabalho, a criação de modelos é facilitada pela informalidade dos modelos semiformais da SysML anotados com o *profile* MARTE, e as análises numéricas são rígidas e precisas por serem realizadas através de modelos analíticos, nesse caso, DSPNs.

1.1 Motivação e Justificativa

A explosão e a popularização dos dispositivos inteligentes fez com que esses dispositivos se tornassem, cada vez mais, essenciais para o estilo de vida das pessoas. Eles conseguiram se tornar mais efetivos e convenientes à comunicação, ao acesso à informação e ao provimento de serviços através da Internet, em qualquer lugar e a qualquer hora. Nesse contexto, o mercado das TICs tem apresentado cada vez mais tendências para oferecer soluções que atendam as platafor-

mas móveis, tais como a criação de *frameworks* específicos para auxílio do desenvolvimento de aplicativos móveis multi-plataformas[1] (ex., *Phonegap*[2] e *Titanium*[3]). O desenvolvimento e criação de *apps* (aplicativos móveis) e sua movimentação através das lojas de aplicativos online (*Google Play, Apple Store, Windows phone marketplace*) também tem crescido em atrativos. Em 2017, espera-se que apenas o mercado de *apps* gere uma receita maior que $77 bilhões de dólares, tornando os aplicativos móveis uma das mais populares ferramentas computacionais para usuários em todo o mundo (MEULEN; RIVERA, 2014). Isto nos mostra uma pré-visualização de um mundo onde os dispositivos móveis são e serão importantes para auxiliar a vida e as atividades das pessoas.

O uso da *mobile cloud computing* a cada ano ganha mais espaço. Espera-se que em 2019 apenas as redes móveis (2G [4], 3G [5], 4G [6]) consumam o equivalente a 24,3 exabytes, quando este número em 2014 foi de apenas 2,5 exabytes. Vale ser ressaltado que, em 2019, 90% desses dados consumidos serão oriundos de uma infraestrutura de computação em nuvem. Além disso, estima-se que o *offloading* de dados seja responsável por 54% dos dados trafegados nos dispositivos móveis (INDEX, 2015).

A energia é um recurso vital para a computação móvel (FLINN; SATYANARAYANAN, 1999). Contudo, dispositivos móveis tem um tamanho reduzido (para serem transportados com mais facilidade) e, devido a este fato, a capacidade de armazenamento de energia nesses tipos de aparelhos é limitada. Dessa maneira, fazer uma melhor gestão desse recurso é de grande importância, visto que pode implicar diretamente no consumo e no melhor aproveitamento desta energia. Aliado a este fator, a realização de mudanças e ajustes nos projetos ainda na sua fase do planejamento já se mostrou mais eficiente e menos custoso, se comparado à realização de alterações na fase de desenvolvimento do projeto (PORTILLO, 2010). Dessa forma, estimativas de desempenho e consumo de energia das aplicações podem ser de grande importância para os projetistas, pois se as análises não apresentarem métricas satisfatórias os projetistas e desenvolvedores poderão realizar alterações nos *softwares* ainda na fase de planejamento, economizando tempo, dinheiro e recursos (CALLOU, 2009).

1.2 Trabalhos Relacionados

Na literatura existem alguns trabalhos com relação direta com a pesquisa aqui apresentada. Os trabalhos de ANDRADE (2014), SHORIN; ZIMMERMANN; MACIEL (2012), ANDRADE (2009) e PEREZ-PALACIN; MIRANDOLA; MERSEGUER (2012) apresentam métodos similares ao proposto neste trabalho para modelagem e análise de sistemas. Entretanto,

[1] O termo se refere ao desenvolvimento de sistemas sob uma plataforma intermediária capaz de ser compreendida por diferentes plataformas/Sistemas Operacionais (SOs)

[2] `http://phonegap.com/`

[3] `http://www.appcelerator.com/titanium/`

[4] Segunda geração de padrões e tecnologias de telefonia móvel (2G)

[5] Terceira geração de padrões e tecnologias de telefonia móvel (3G)

[6] Quarta geração de padrões e tecnologias de telefonia móvel (4G)

nenhum deles faz uso de mapeamentos de diagramas SysML em DSPNs para analisar questões de desempenho, consumo de energia e disponibilidade em aplicações presentes em dispositivos móveis. No trabalho de ANDRADE (2014), o autor se utiliza da metodologia desenvolvida para avaliação de métricas relacionadas à disponibilidade. Já os trabalhos de SHORIN; ZIMMERMANN; MACIEL (2012) e ANDRADE (2009) utilizam a metodologia para avaliação de métricas de desempenho e consumo de energia em sistemas embarcados. O trabalho de PEREZ-PALACIN; MIRANDOLA; MERSEGUER (2012) utiliza diagramas da UML e SPNs para avaliar e analisar o consumo de energia de grandes infraestruturas computacionais. Porém, não é utilizado mapeamento de modelos UML em redes de Petri. Desta forma, podemos classificar os trabalhos relacionados em três grandes categorias: (i) Trabalhos que se utilizam do mapeamento de modelos semiformais em modelos formais para realização de análises em sistemas, (ii) trabalhos que usam de modelos formais para análise de consumo de energia, e (iii) trabalhos que apresentam técnicas/ferramentas para economia de energia em dispositivos móveis.

ANDRADE (2014) apresenta um *framework* que permite mapear diagramas da SysML anotados com o *profile* MARTE em DSPNs. O trabalho foca na modelagem e análise de sistemas distribuídos, observando métricas relacionadas à disponibilidade, afim de maximizar o uso dos recursos computacionais com o menor custo atrelado possível. Dessa forma, o trabalho também fornece meios para se estudar os mecanismos de tratamento de interrupções, bem como as infraestruturas dos sistemas distribuídos de forma simples e sem a necessidade de conhecimento especializado em modelagem estocástica. Ademais, o autor também apresenta o OpenMADS (ANDRADE et al., 2013), uma ferramenta que implementa o *framework* proposto. A ferramenta possui integração com as ferramentas ASTRO/Mercury (SILVA, 2013) e TimeNET (ZIMMERMANN; KNOKE, 2007), onde é possível realizar análises estocásticas dos modelos DSPNs obtidos automaticamente pelo OpenMADS. Vale ressaltar que esta dissertação se baseia no trabalho deste autor, adotando seu *framework* como base para construção de uma abordagem que utiliza mapeamento de diagramas da SysML em DSPNs, contudo, com enfoque para analisar questões relacionadas ao desempenho, consumo de energia e disponibilidade e aplicações móveis. A diferença entre esse trabalho e a dissertação aqui apresentada é percebida na abordagem proposta. A abordagem contêm menos passos, diferentes estereótipos e valores marcados do MARTE foram usados, além da mudança de foco das análises.

No mesmo contexto, de abordagens que lidam com o mapeamento de modelos semiformais para modelos analíticos, SHORIN; ZIMMERMANN; MACIEL (2012) propõem uma metodologia para modelagem e análise de desempenho e consumo de energia de sistemas embarcados. Os autores utilizam diagramas de estados da UML junto com o *profile* MARTE para obtenção de SPNs (MARSAN, 1990). Uma vez obtido os modelos SPNs, então análises são realizadas para obtenção do valor estimado do consumo de energia. Também são definidas regras de mapeamento dos elementos da UML para SPNs. Ademais, apenas um estudo de caso é apresentado para demonstrar a aplicabilidade da metodologia que estima o consumo de

energia de um microcontrolador. Esse trabalho não tem como foco a obtenção de métricas de disponibilidade para o sistema modelado. A abordagem apresentada nesta dissertação utiliza dois tipos de diagramas da SysML e DSPNs para isto. Por outro lado, no trabalho apresentado por SHORIN; ZIMMERMANN; MACIEL (2012) é utilizado apenas o diagrama de estados da UML para apresentar estimativas das métricas de desempenho e consumo de energia, não apresentando métricas relacionadas à disponibilidade.

No trabalho realizado por ANDRADE (2009), o autor também propõe o mapeamento de modelos semiformais em modelos formais. Porém, essa abordagem mapeia os diagramas comportamentais da SysML em uma Rede de Petri Temporizada com restrição de Energia (ETPN) (TAVARES et al., 2008). O trabalho foca na avaliação de métricas de desempenho e consumo de energia em sistemas embarcados de tempo-real críticos. As restrições de tempo e anotações de energia são representadas pelo profile MARTE. O método proposto consiste, primeiramente, na derivação dos elementos básicos de cada diagrama da SysML em modelos ETPN individuais. Após a obtenção dos modelos ETPN, análises e verificações são executadas. Além disso, uma metodologia de avaliação de desempenho das especificações de sistemas críticos é proposta com o intuito de auxiliar o processo de modelagem e avaliação. As principais diferenças estão no autor utilizar um tipo diferente de redes de Petri e focar na análise de sistemas embarcados de tempo-real críticos. Além disso, diferentes diagramas da SysML foram utilizados para atender as necessidades de modelagem.

Já no trabalho apresentado por PEREZ-PALACIN; MIRANDOLA; MERSEGUER (2012), os autores propõem um *framework* auto adaptável para ajudar na redução de consumo de energia de infraestruturas computacionais como *data centers*. A abordagem se baseia no provimento de energia proporcional a utilização de recursos, ou seja, leva-se em conta a carga de trabalho que está sendo utilizada para prover os recursos necessários. Para isto, é considerado tanto o número de servidores ligados quanto as frequências de operação da Unidade de Processamento Central (CPU) dos mesmos. Os autores também utilizam a transformação de diagramas UML em modelos SPNs para modelar as funcionalidades do *framework* e analisar os resultados de sua utilização, sem a necessidade da implementação real, que seria custosa e demorada. Foi exibido um exemplo da aplicação do *framework* através de modelos, onde o plano mostrado pode economizar até 50% de energia se comparado ao cenário sem a adoção do *framework*. A distinção entre nosso trabalho e o apresentado por PEREZ-PALACIN; MIRANDOLA; MERSEGUER (2012) está no uso de mapeamentos para obtenção das redes de Petri e no foco dos estudos, onde o autor utilizou diagramas UML e modelos SPNs apenas para validar o uso do *framework* que seria desenvolvido. Nessa abordagem vale frisar que os pesquisadores/projetistas precisavam ter conhecimento acerca de SPNs para criação dos modelos, já que não foi utilizado nenhum tipo de mapeamento.

Em OLIVEIRA et al. (2013), os autores apresentam um estudo de disponibilidade para *mobile cloud*. Essa abordagem faz uso da combinação dos modelos de Diagrama de Blocos de Confiabilidade (RBD) e redes de Petri para avaliar o impacto da comunicação via Rede sem-

fio (WiFi) sobre o consumo de energia dos dispositivos móveis e a disponibilidade do sistema. Os autores apresentam um modelo em rede de Petri para representar a descarga da bateria do dispositivo móvel por uma aplicação que utiliza WiFi e 3G como tipos de conexões. No estudo de caso e nas análises de cenários realizadas, os resultados mostram que o uso do protocolo 3G causa uma diminuição na taxa de disponibilidade e é o grande responsável pelo descarregamento da bateria do aparelho. Entretanto, se for combinado com o protocolo WiFi, provendo uma redundância de comunicação, a disponibilidade e o tempo de vida da bateria dos dispositivos apresentam melhores resultados. Apesar de oferecer modelos estocásticos para realização de análises de disponibilidade e consumo de energia, nesse trabalho, a análise do consumo de energia o estudo foca na utilização de conexões redes sem fio. Já na abordagem apresentada no nosso trabalho, as análises de consumo de energia podem ser realizadas independente de usarem ou não conexões de rede.

Já CALLOU (2009), propõe uma metodologia para ser aplicada nas fases iniciais de projeto, a fim de prover suporte às decisões relativas ao consumo de energia e desempenho de sistemas embarcados. Nessa abordagem é proposta a utilização de modelos em redes de Petri coloridas (JENSEN, 1995) para realização da análise quantitativa e verificações de propriedades. Os resultados apresentados dos estudos de caso mostram que as estimativas realizadas através dos modelos chegam muito próximos aos valores obtidos através de experimentos em um hardware real. Além disso, o autor também apresenta o *software* ALUPAS, responsável por estimar o consumo de energia e o tempo de execução dos sistemas embarcados. No trabalho de CALLOU (2009), o foco da realização de análises está relacionado com sistemas embarcados, além disso, não são apresentadas métricas de disponibilidade do sistema. Isto diferencia nosso trabalho do trabalho apresentado por este autor.

Utilizando um modelo matemático, o trabalho proposto por HARJULA; KASSINEN; YLIANTTILA (2012) também busca estudar o consumo de energia em dispositivos móveis que utilizam conexões de redes 3G e Rede de área local sem-fio (WLAN). O modelo faz distinção entre os elementos de sinalização (transmissão de pacotes de dados únicos) e transferência de arquivos (transmissão de pacotes de dados em série) para estimar o consumo de energia. Os resultados do trabalho mostram que, se comparado a experimentos em um dispositivo real, o modelo apresenta uma precisão de 94%-97% na estimativa de consumo de energia para transmissão de pacotes de tamanhos fixos no cenário de sinalização. Contudo, se for considerado um tamanho variável de pacotes, a margem de erro da estimativa do modelo é de 14%-21%. Já para o cenário de transmissão de arquivos, o modelo apresenta uma margem de erro menor que 1%. Nossa pesquisa se diferencia desse trabalho por apresentar modelos que são de fácil entendimento para projetistas. A utilização de modelos matemáticos complexos exige um bom conhecimento na área, o que pode dificultar a adoção da abordagem.

Seguindo a mesma temática, BALASUBRAMANIAN; BALASUBRAMANIAN; VENKATARAMANI (2009) apresentam um estudo de medição de consumo de energia em celulares, levando em conta três conexões de rede: 3G, 2G e WiFi. A partir desse estudo, os autores desen-

volveram um modelo de consumo de energia para cada tecnologia de conexão de rede. Utilizando os modelos construídos, os autores desenvolveram o *TailEnder*, um protocolo capaz de reduzir o consumo de energia de aplicações móveis comuns. O algoritmo de agendamento utilizado pelo protocolo tenta reduzir o consumo energético causando *delays* em tarefas que podem aguardar para serem realizadas. Os resultados mostrados através de experimentos mostram que com o uso do protocolo aplicações podem fazer *download* de 60% a mais de dados oriundos de *feeds* de notícias e realizar 50% a mais de buscas na *web*, se comparados com o uso sem o protocolo *TailEnder*. Além disso, resultados de simulações mostraram que a utilização do protocolo pode reduzir em 35% o consumo de energia para aplicações de email, 52% para aplicações de *feed* de notícias e 40% para aplicações de buscas na *web*. Apesar de também auxiliar na redução do consumo de energia, nesse trabalho é necessário a implementação do protocolo nas aplicações, o que algumas vezes pode ser complicado e demorado. Esta é a principal diferença para nosso trabalho, a utilização de modelos pode ser mais rápida e ter um custo menor. Nesta dissertação, o foco do trabalho é auxiliar o melhor planejamento de para a construção mais eficiente de aplicações.

Por fim, DOLEZAL; BECVAR (2014) descrevem uma metodologia e uma ferramenta (I.T.E.M.) desenvolvida para medição de consumo de energia de dispositivos móveis. A ferramenta desenvolvida, baseada na metodologia, auxilia na medição de energia gasta pelos componentes de smartphones (CPU, tela, Operações de escrita e leitura e auto falantes). Com base nas medições realizadas, os autores também propõem modelos matemáticos para ajudar a estimar o consumo de energia dos dispositivos móveis sem a necessidade da realização de medições em tempo real. Contudo, os autores afirmam que os modelos matemáticos ainda precisam ser melhores investigados para fornecer uma melhor comparação com longos experimentos. O trabalho de DOLEZAL; BECVAR (2014) apresenta uma ferramenta e uma metodologia para ajudar nas análises de consumo de energia de aplicações. Contudo, os autores usam modelos matemáticos, sendo necessário conhecimentos mais especializados para seu uso. No nosso trabalho, são utilizados modelos semiformais já difundidos no mercado que possuem uma fácil notação.

Para uma melhor visualização e comparação entre as características dos trabalhos relacionados, é apresentada a Tabela 1.1. Nessa tabela estão assinaladas as macro-características de cada trabalho relacionado citado, além das características do trabalho apresentado nesta dissertação.

Tabela 1.1: Comparação de características entre os trabalhos relacionados.

	Utilização de Modelos			Análise de Métricas			Contexto de sistemas		Ferramenta		
Trabalhos	Semiformais	Formais	Mapeamento	Desempenho	Energia	Disponibilidade	Embarcados	Infra. computacional	Software	Algoritmo	Técnica/Abordagem
ANDRADE (2014)	X	X	X			X		X	X		X
SHORIN; ZIMMERMANN; MACIEL (2012)	X	X	X	X	X		X				X
ANDRADE (2009)	X	X	X	X	X		X				X
PEREZ-PALACIN; MIRANDOLA; MERSEGUER (2012)	X	X		X	X			X			
OLIVEIRA et al. (2013)		X			X	X	X				X
CALLOU (2009)		X		X	X		X		X		X
HARJULA; KASSINEN; YLIANTTILA (2012)		X			X		X			X	X
BALASUBRAMANIAN; BALASUBRAMANIAN; VENKATARAMANI (2009)		X			X		X			X	
DOLEZAL; BECVAR (2014)		X			X		X				X
Este Trabalho	X	X	X	X	X	X	X				X

1.3 Objetivos

O objetivo principal deste trabalho é propor uma abordagem para auxiliar projetistas de sistemas na avaliação de desempenho, consumo de energia e disponibilidade de aplicações móveis. Para isto, é combinada a utilização de modelos semiformais com modelos formais, fazendo o mapeamento de diagramas da SysML em Redes de Petri Estocásticas e Determinísticas (DSPNs). Desta forma, facilitando a construção destes modelos analíticos por projetistas que possuem pouca *expertise* em modelagem estocástica, visando auxiliá-los nos estudos de métricas relacionadas a desempenho, consumo de energia e disponibilidade das aplicações móveis a serem desenvolvidas.

Os objetivos específicos deste trabalho são:

- desenvolver uma abordagem que auxiliem projetistas a avaliarem métricas de desempenho, consumo de energia e disponibilidade de aplicações móveis;

- apresentar um método de mapeamento dos diagramas da SysML em modelos DSPN para análises de desempenho, consumo de energia e disponibilidade;

1.4 Organização da Dissertação

No Capítulo 2 são introduzidos os conceitos fundamentais do trabalho: consumo de energia e análise de desempenho, fundamentos de *mobile cloud computing*, SysML, MARTE, redes de Petri e RBD. O Capítulo 3 apresenta a abordagem desenvolvida para modelagem e análise de desempenho e consumo de energia de aplicações móveis, bem como as regras de mapeamento utilizadas. No Capítulo 4 é apresentado o método utilizado para a realização das medições de consumo de energia na aplicação móvel utilizada, além do cenário utilizado para o estudo. O Capítulo 5 mostra a validação dos modelos utilizados na modelagem do cenário base estudado e outros estudos de casos nos quais foram aplicados a abordagem apresentada. Por fim, o Capítulo 6 conclui o trabalho e apresenta as contribuições, limitações e os possíveis trabalhos futuros.

2

Fundamentação Teórica

Education is the most powerful weapon which you can use to change the world.

—NELSON MANDELA

Este capítulo apresenta os principais conceitos sobre os temas abordados para um melhor entendimento deste trabalho. Primeiramente, conceitos sobre avaliação de desempenho e consumo de energia são introduzidos. Depois são apresentados os fundamentos sobre *Mobile Cloud Computing* (MCC). Posteriormente são mostrados os conceitos sobre a *System Modelling Language* (SysML) e os diagramas utilizados na pequisa, além do *Profile for Modeling and Analysis of Real-time and Embedded systems* (MARTE). Depois são abordados os principais conceitos acerca das redes Petri. Por último, conceitos acerca do Diagrama de blocos de Confiabilidade.

2.1 Avaliação de desempenho e consumo de energia

A energia é um recurso vital para a computação móvel (FLINN; SATYANARAYANAN, 1999). Os desafios relacionados ao consumo de energia podem ser divididos em: (i)análise e (ii) otimização (YEAP, 1998). Os problemas de análise estão relacionados a precisão nas estimativas de consumo de energia. Técnicas de análises se diferenciam quanto a precisão e eficiência. Esses dois fatores dependem da quantidade de informação a ser utilizada para processar as análises. Por outro lado, a otimização automática requer uma *engine*, capaz de fazer avaliação de diferentes cenários. A otimização manual, além de demandar mais tempo, necessita de ferramentas para estimar o consumo de energia de cada cenário a ser analisado (CALLOU, 2009).

Poder estimar o desempenho e o consumo de energia das aplicações pode ser de grande importância para os projetistas, pois se as análises não apresentarem resultados satisfatórios, eles poderão realizar alterações nos *softwares* ainda na fase de planejamento, economizando tempo, dinheiro e recursos (CALLOU, 2009).

A avaliação de desempenho de sistemas computacionais consiste em um conjunto de

critérios e técnicas classificadas como: (i) baseadas em medição e (ii) baseadas em modelagem. As técnicas baseadas em modelagem podem ser classificadas como analíticas e de simulação (JAIN, 1991). A avaliação de desempenho por técnicas de modelagem costumam ocorrer nos primeiros estágios de desenvolvimento de projetos, onde não é possível a realização de medições no sistema (JOHN; EECKHOUT, 2006). A Figura 2.1 ilustra a classificação, citando algumas técnicas.

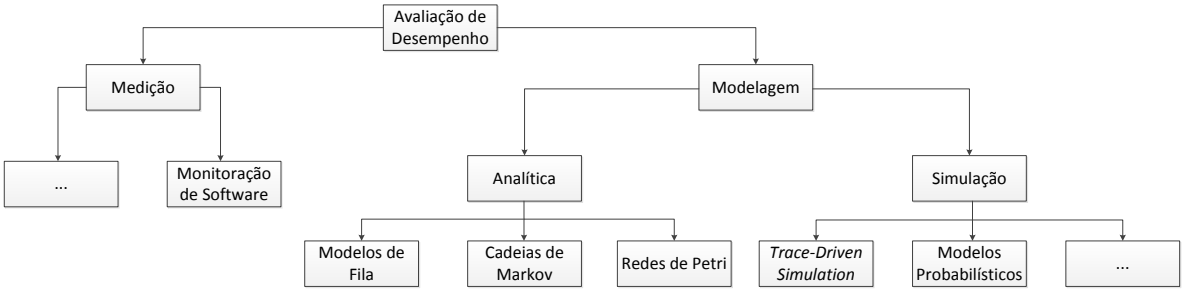


Figura 2.1: Classificação das técnicas de análises de desempenho.

Adaptado: JOHN; EECKHOUT (2006).

Na modelagem analítica, o comportamento do sistema é descrito através de funções matemáticas. Geralmente os modelos consideram parâmetros específicos de um sistema, mas que são facilmente adaptados para outros sistemas. Os modelos analíticos permitem uma análise profunda em relação as métricas de interesse, uma vez que são definidos sob fortes fundamentos matemáticos. Esse tipo de modelagem apresenta um menor custo, se comparado a outras. Valores obtidos através de medições experimentais do sistema podem ser comparados aos resultados apresentados pelos modelos, afim de fazer a validação do modelo e uma melhor análise de cenários.

2.2 Fundamentos sobre *Mobile Cloud Computing*

O crescimento conjunto da computação móvel e da computação em nuvem fez com que em meados de 2007 fosse introduzido o paradigma da *Mobile Cloud Computing* (MCC) (DINH et al., 2011). Seu principal objetivo é o provimento de recursos computacionais da nuvem para aumentar o poder das tarefas executadas pelos dispositivos móveis. A seguir são apresentados conceitos acerca de computação em nuvem e *mobile cloud computing*.

2.2.1 Computação em nuvem

O Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST) define a computação em nuvem como (tradução livre):

> *"Um modelo que fornece conveniência e provê acesso sob demanda, através da rede, a um conjunto de recursos computacionais compartilhados e configuráveis (redes,*

> *servidores, armazenamento, aplicações e serviços), que podem ser rapidamente disponibilizados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços."(BADGER et al., 2012)*

Este modelo permite que usuários tenham acesso aos recursos computacionais de forma elástica, sob demanda e a um baixo custo, entregues de maneira semelhante a serviços tradicionais como água, gás, eletricidade e telefonia. Isto significa que os usuários podem solicitar estes recursos sempre que necessário, e serão cobrados apenas pela utilização deles durante o período de tempo que estes recursos foram disponibilizados (DINH et al., 2011). A computação em nuvem é um paradigma computacional recente, mas que foi construído com base em tecnologias existentes como virtualização, *grid computing* e *utility computing*. Os recursos computacionais de uma *grid* são provisionados para os clientes como Máquinas Virtuais (VMs) através da virtualização de uma infraestrutura de *datacenter*, de forma que cada cliente só paga pela quantidade de recursos que consumir, em vez de pagar uma taxa fixa (*utility computing*) (OLIVEIRA, 2014).

O NIST (BADGER et al., 2012) define três tipos de modelos de negócio para oferta de serviços da computação em nuvem: *Software* como Serviço (SaaS), Plataforma como Serviço (PaaS) e Infraestrutura como Serviço (IaaS).

- **SaaS** O consumidor utiliza a infraestrutura provida pelo fornecedor apenas para consumir aplicações presentes na nuvem. As aplicações podem ser acessíveis a partir de diversos dispositivos clientes, seja por um *browser* ou um aplicativo, por exemplo. O consumidor não pode administrar nem controlar a infraestrutura da nuvem.

- **PaaS** O consumidor utilizada a infraestrutura da nuvem utilizando bibliotecas de linguagens de programação suportadas pelo provedor para controlar e utilizar os recursos da nuvem na aplicação desenvolvida. Contudo, o consumidor não pode administrar e nem controlar a rede, servidores e sistemas operacionais, mas tem controle sobre os aplicativos implementados e possivelmente sobre as configurações para o ambiente de hospedagem de aplicativos.

- **IaaS** O consumidor tem acesso ao provisionamento de processamento, armazenamento, redes e outros recursos de computação fundamentais, onde é possível implementar e executar *softwares*, que podem incluir desde sistemas operacionais a aplicações. Porém, o consumidor não pode administrar nem controlar a infraestrutura de nuvem subjacente, mas tem um controle limitado de componentes de rede selecionados (por exemplo, *firewalls* do *host*).

Outros modelos também estão ganhando espaço e se tornando mais populares. Dentre eles, podemos destacar o **Dados como Serviço (DaaS)** e o ***Mobile Backend* como Serviço (MBaaS)**. O primeiro pode ser definido como um modelo para prover o armazenamento, gestão e fornecimento de dados, entregues em formato imediatamente consumíveis para os usuários de

negócio de empresas, como serviço. Este modelo emerge como resposta natural ao o crescente volume e variedade de dados utilizados atualmente, além das habilidades necessárias para lidar com estes dados (PRINGLE, 2014). O segundo, vêm ganhando impulso com o paradigma da *Mobile Cloud Computing* (MCC). O MBaaS é uma abordagem para prover conectividade a aplicações *mobile* (na maioria dos casos) e *Web* à serviços disponibilizados na nuvem. Ajudando a prover funcionalidades como, gerenciamento de usuários, *push notifications*, integração com redes sociais e outras funcionalidades comuns as utilizadas pelas aplicações móveis atuais. Estes serviços são disponibilizados através de um *Software Development Kit* (SDK) ou uma *Application Programming Interface* (API) (API Evangelist, 2015).

Sob os aspectos de infraestrutura, o NIST (BADGER et al., 2012) define quatro tipos de nuvem:

- **Privada.** A infraestrutura de nuvem é fornecida para uso exclusivo de uma única organização (por exemplo, uma empresa). Ela pode ser controlada, gerenciada e operada pela organização ou um terceiro.
- **Comunitária.** A infraestrutura de nuvem é fornecida para uma comunidade específica onde os consumidores têm preocupações comuns (por exemplo, grupos de pesquisa). Ela pode ser controlada, gerenciada e operada por uma ou mais organizações da comunidade ou um terceiro.
- **Pública.** A infraestrutura de nuvem é fornecida para uso aberto ao público em geral. Ela pode ser controlada, gerenciada e operada por organização empresarial, acadêmica ou governamental. O fornecedor da nuvem determina as regras de utilização.
- **Híbrida.** A infraestrutura de nuvem é uma composição de duas ou mais infraestruturas de nuvem distintas (privada, comunitária ou pública), mas possuem tecnologia padronizada que permita a portabilidade de dados (por exemplo, balanceamento de carga entre nuvens).

2.2.2 *Mobile Cloud Computing*

Há diferentes formas de se entender a definição de *Mobile Cloud Computing* (MCC). Neste trabalho, foi adotada a definição de (KOVACHEV; CAO; KLAMMA, 2010):

> *"Mobile cloud computing é um modelo para aumento elástico transparente dos recursos do dispositivo móvel através do acesso ubíquo a redes sem fio para o armazenamento em nuvem e acesso a recursos computacionais, com ajuste dinâmico de contexto para offloading, afim de mudar condições de operações, preservando ao mesmo tempo as capacidades de detecção e interação dos dispositivos móveis."(KOVACHEV; CAO; KLAMMA, 2010)*

Mesmo com a evolução e melhoramento do *hardware* do dispositivo móvel e das redes móveis, os dispositivos móveis serão sempre pobres em recursos, menos seguros, com conectividade instável, e com menos energia, uma vez que são alimentados por bateria (KOVACHEV; CAO; KLAMMA, 2010). Nesse sentindo, ABOLFAZLI et al. (2014) aponta as principais motivações para a necessidade de aumento dos recursos computacionais dos dispositivos móveis (ver Figura 2.2): *(i) Poder de processamento*: espera-se que os dispositivos móveis tenham um alto poder de processamento, similar aos *desktops* para processar uma grande quantidade de informações, com o objetivo de oferecer uma melhor experiência de uso para o usuário. *(ii) Recursos energéticos*: energia é um recurso que depende de fontes externas. A capacidade da bateria aumenta de 5 a 10% por ano (ROBINSON, 2009). Entretanto, os fabricantes dos dispositivos trabalham para conseguir deixar os dispositivos mais leves e compactos, o que impede a adoção de baterias mais volumosas e de longa duração. *(iii) Armazenamento Local*: o aumento de aplicações capazes de gerar conteúdo digital, como imagens, vídeos e músicas é muito grande, e o dispositivo móvel possui armazenamento limitado devido ao seu tamanho reduzido. *(iv) Capacidade de visualização*: disponibilizar uma visualização eficiente de dados em uma tela pequena não é uma tarefa simples, quando se tem tecnologias diferentes e uma fonte de energia reduzida. São necessárias técnicas baseadas em softwares ao invés de *hardware* para aumentar virtualmente a área de apresentação dos objetos. *(v) Segurança, Privacidade e Segurança de dados*: segurança e privacidade sempre são desafios complexos para sistemas, e executar algoritmos de criptografia complexos é provavelmente inviável devido as deficiências do poder computacional destes dispositivos. E a segurança dos dados apresenta mais desafios, uma vez que existe uma maior probabilidade de um dano físico, mal funcionamento do *hardware*, perda ou roubo do dispositivo.

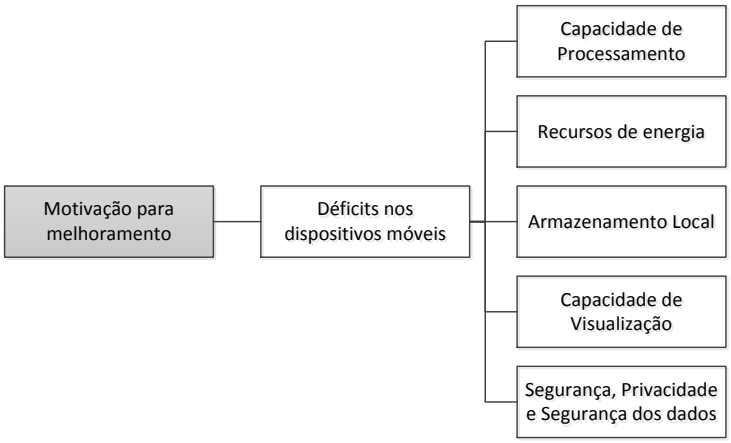


Figura 2.2: Déficits dos dispositivos móveis.

Adaptado: ABOLFAZLI et al. (2014).

Dessa forma, podemos observar que o paradigma da MCC tem por objetivo otimizar os recursos dos dispositivos móveis, utilizando a capacidade de processamento e armazenamento

de uma nuvem para melhorar o desempenho e aumentar a capacidade de operação do dispositivo. Além disso, este paradigma pode ajudar no aumento do tempo de vida da bateria dos dispositivos móveis, uma vez que tarefas podem ser transferidas para os servidores na nuvem (*offloading* (RUDENKO et al., 1998)) mais facilmente (DINH et al., 2011). Ademais, a MCC pode permitir o provisionamento de aplicações móveis mais inteligentes, que estendam as capacidades cognitivas de usuários, tais como: reconhecimento de voz, processamento de linguagem natural, visão computacional, aprendizado de máquina, realidade aumentada, planejamento e tomada de decisão (SATYANARAYANAN et al., 2009). E enquanto os dispositivos móveis não tiverem um poder de processamento ou memória para suportar o grande número de dados, a computação em nuvem parece ser a solução ideal para os usuários destes dispositivos (SHARMA; KUMAR; TRIVEDI, 2013).

Com o surgimento da *Mobile Cloud Computing* (MCC), o modelo de serviço de MBaaS passou ser oferecido aos usuários, através do modelo *on-demand* (sob demanda), disponibilizando facilidades na construção de aplicações móveis. Dentre os principais MBaaS atuais podemos citar o Kinvey[1], Anypresence[2], Appcelerator [3], Sencha [4] e Parse[5].

Na Figura 2.3 é apresentado um exemplo de arquitetura de uma MCC. DINH et al. (2011) define os seguintes componentes em uma MCC: (i) Os dispositivos móveis - são conectados a redes móveis através de (ii) operadores de redes (satélite, *access point* ou estação de transmissão de sinal). Quando informações são solicitadas pelos usuários, a requisição é transmitida para uma central de processamento - que é conectada aos servidores que proveem os serviços de autenticação, autorização e gerenciamento de usuários. Depois de processadas as informações chegam a (iii) nuvem através da (iv) Internet. Na nuvem, os (v) controladores - processam a requisição para prover o serviço requisitado aos usuários. Vale destacar, que o autor afirma que a arquitetura pode ser diferente, dependendo do contexto.

2.3 SysML

A SysML *"é uma linguagem de modelagem visual que estende a UML 2 afim de fornecer suporte a especificação, análise, design, verificação e validação de sistemas complexos que incluem componentes de hardware, software, dados, pessoal, procedimentos e instalações"*(HAUSE, 2006). Além de fornecer um padrão de modelagem, a SysML busca o aperfeiçoamento da qualidade do sistema, assegurando a troca de informações e diminuindo a distância semântica entre a engenharia de sistemas, software, e outras áreas da engenharia (SILVA, 2006). Ela foi projetada para ser utilizada com diferentes metodologias, incluindo análise estrutural, orientação a objetos e outros (HAUSE, 2006).

[1] http://kinvey.com/
[2] http://www.anypresence.com/
[3] http://www.appcelerator.com/
[4] http://www.sencha.com/
[5] https://parse.com/

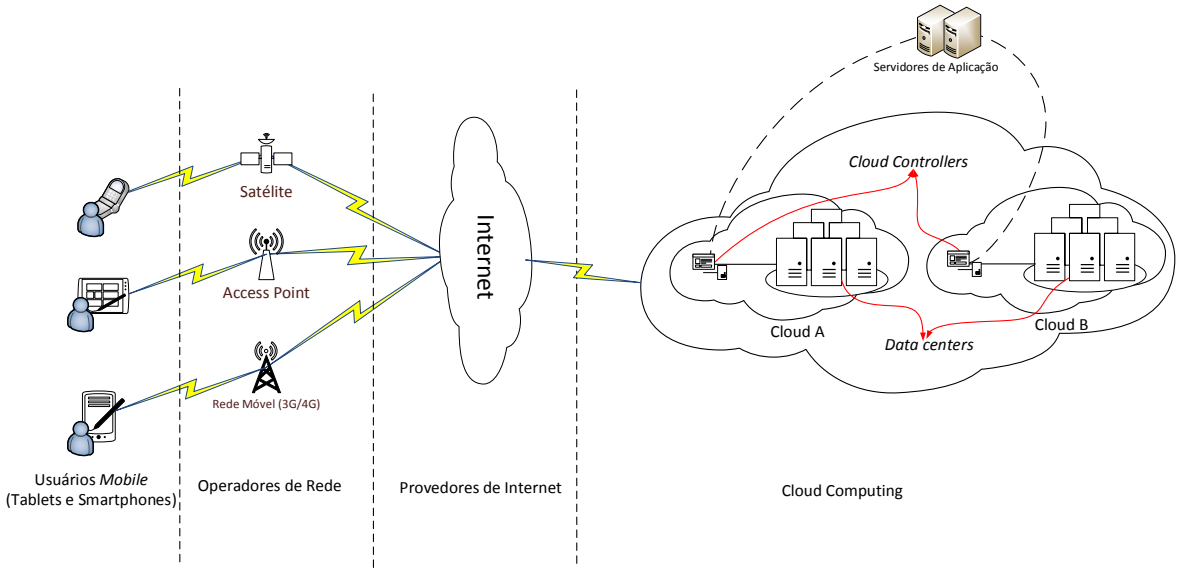


Figura 2.3: Arquitetura de uma MCC.

Adaptado: DINH et al. (2011).

Definida com base na UML 2 (OMG, 2004), a SysML utiliza em parte sua sintaxe e semântica. A UML é bastante difundida na indústria como linguagem para modelagem de sistemas. Contudo, quando é necessária uma abordagem multidisciplinar (*software*, eletrônica, mecânica), ela se faz limitada. Nestes casos, a SysML é mais aceita para cumprir este papel, sendo projetada para permitir extensões que suportam domínios especializados. Ela inclui os seus próprios mecanismos de extensão, chamados *estereótipos*, que permitem classificar os elementos que possuem algo em comum (FRIEDENTHAL; MOORE; STEINER, 2008). Os *estereótipos* utilizados neste trabalho são descritos no decorrer da dissertação.

Nove tipos de diagramas compõem a SysML, alguns importados sem modificações da UML 2, outros adicionados ou modificados para atender às necessidades da engenharia de sistemas (ver Figura 2.4). Os diagramas podem ser divididos em três partes: estrutural, comportamental e genérico, como descritos a seguir (HAUSE, 2006):

- **Estrutural.** Estes diagramas da SysML representam a estrutura do sistema, os elementos conceituais ou físicos. O diagrama de definição descreve a hierarquia do sistema e a classificação dos sistemas ou componentes. O diagrama de bloco interno descreve a estrutura interna do sistema. O diagrama de pacote é utilizado para organização do modelo. O diagrama paramétrico é usado para expressar as restrições impostas pelos sistemas.

- **Comportamental.** Os diagramas comportamentais da SysML representam o comportamento do sistema no tempo e no espaço. O diagrama de caso de uso provê uma descrição do sistema em alto nível. O diagrama de atividades descreve o fluxo de dados e controle entre as ações. O diagrama de sequência apresenta a interação

entre as entidades do sistema. O diagrama de estados descreve o comportamento do sistema, mostrando as transições de estado, além das ações que o sistema ou seus elementos realizam em resposta aos eventos. Tipicamente, esse diagrama é utilizado para representar o ciclo de vida de um bloco (SILVA, 2006).

- **Genérico *(Cross-cutting constructs)*.** Estes diagramas podem ser aplicados tanto nas partes estruturais quanto nas comportamentais. O diagrama de requisitos representa um requisito baseado em texto (definido textualmente). O relacionamento de alocação representa as relações que mapeiam um elemento do modelo em outro. A SysML suporta várias formas de alocações entre os diferentes elementos do modelo, incluindo a alocação comportamental, a estrutural e a de propriedades (SILVA, 2006).

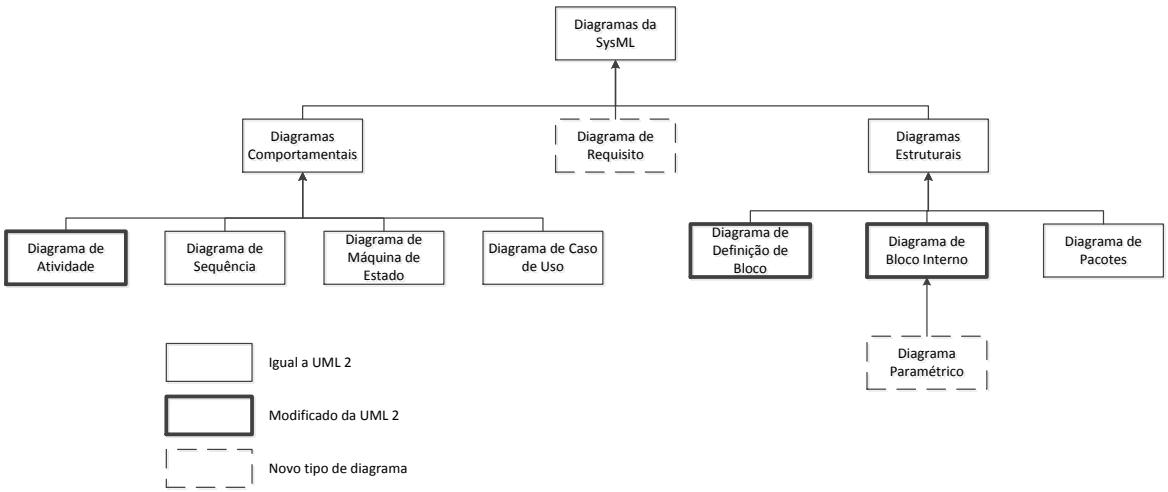


Figura 2.4: Estrutura da SysML.

Adaptado: HAUSE (2006).

Quanto à modelagem visual, a SysML possui um quadro (*frame*) padrão para todos os diagramas (ver Figura 2.5). O quadro é composto por cabeçalho e área do conteúdo. Ele delimita o espaço para modelagem dos diagramas. O cabeçalho deve indicar o tipo do diagrama e o nome do diagrama. Na área do conteúdo é encontrado o modelo propriamente dito.

2.3.1 SysML-IBD

Diagrama de Bloco Interno da SysML (SysML-IBD) é utilizado para representação das configurações estáticas do sistema a ser modelado, como componentes de infraestrutura ou *hardwares*. Geralmente estes componentes possuem relacionamentos entre si. Na Figura 2.6, é mostrado um exemplo de um SysML-IBD. Ele é composto pelos elementos descritos a seguir (TEAM, 2012):

- **Bloco.** É utilizado para representação de qualquer elemento de um sistema.

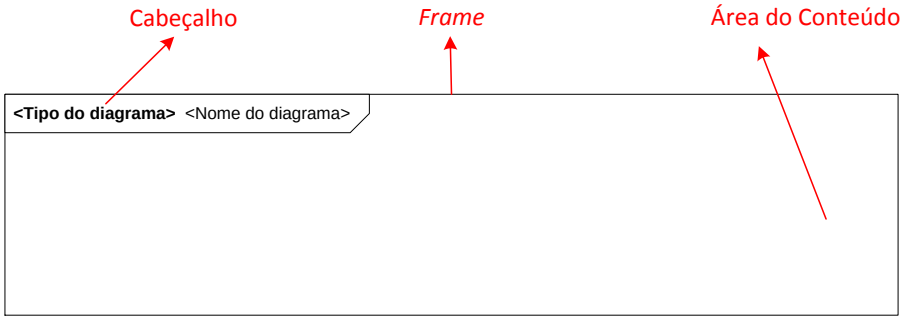


Figura 2.5: Quadro para modelagem dos Diagramas da SysML.

- **Conector.** É usado para indicar comunicação entre blocos, podendo especificar entrada ou saída de dados.
- **Fluxo.** Define o tipo de dado que é trafegado entre blocos.
- **Porta.** Descreve o ponto de interação entre blocos, especificando os tipos de serviços que o bloco disponibiliza ou aceita.

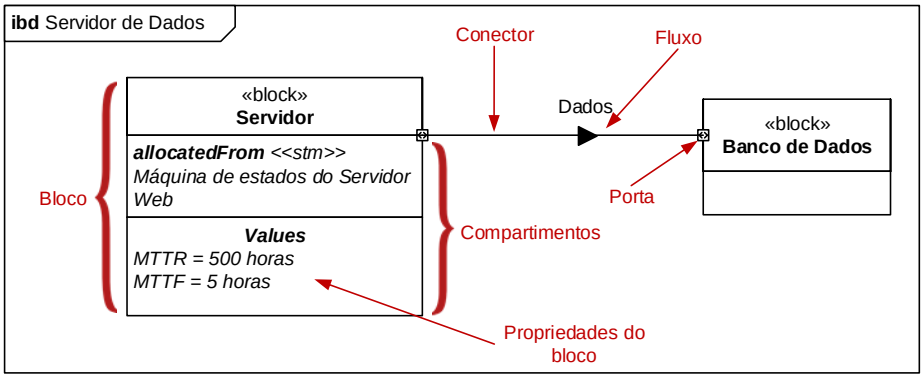


Figura 2.6: Exemplo de um SysML-IBD.

Adaptado: ANDRADE (2014).

O bloco é a unidade principal da SysML. Ele pode ser utilizado para representação de qualquer nível de hierarquia de sistema, incluindo alto nível, subsistemas, ou componentes lógicos e físicos (HAUSE, 2006). Blocos pode se conectar com outros blocos a fim de formar estruturas mais complexas. Além disso, múltiplos compartimentos podem ser utilizados para descrever as características do bloco, tais como propriedades, operações, restrições, alocações, e requerimento que o bloco satisfaz (TEAM, 2012).

2.3.2 SysML-STM

O Diagrama de Estados da SysML (SysML-STM) é usado para representar o comportamento dinâmico de um elemento ou de um sistema. Nele são indicadas as mudanças de estados e as ações do sistema, ou de parte dele, em resposta a ocorrência de eventos. O diagrama de estados foi importado da UML para SysML sem modificações. Na Figura 2.7 é exibido um exemplo de um SysML-STM. Ele é composto pelos seguintes elementos (TEAM, 2012):

- **Decisão.** Estrutura de controle de fluxo usada para controlar desvios no fluxo de execução do sistema modelado. É composto de condições *booleanas* e cada condição, quando satisfeita, dispara uma transição correspondente.
- **Estado.** Representa a situação em que um elemento do sistema (ou o próprio sistema) se encontra em determinado momento. Existem três tipos de estados: estado simples, estado composto e sub-máquina. Estados simples são os quais não possuem subestados; Estados compostos são estados que contêm dois ou mais estados. As sub-máquinas são estados compostos, nos quais os subestados são descritos em outro diagrama. Além disso, estados podem conter atividades internas (*entry*, *do* e *exit*) e transições internas (GUEDES, 2008).
- **Estado Inicial.** Determina o início de um SysML-STM ou de um estado composto.
- **Estado Final.** Determina o fim de um SysML-STM ou de um estado composto.
- **Transição.** Representa um evento que causa uma mudança de estado, de um elemento ou de um sistema. Uma transição é representada por uma seta ligando dois estados.

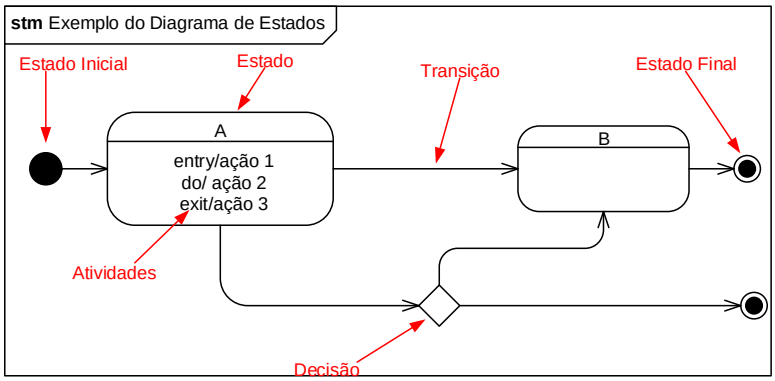


Figura 2.7: Exemplo de um SysML-STM.

Adaptado: ANDRADE (2014)

.

2.4 MARTE

Profile for Modeling and Analysis of Real-time and Embedded systems (MARTE) (OMG., 2011) é um *profile* da UML para auxiliar a criação de modelos que possam ser usados para análises quantitativas de características de sistemas embarcados de tempo-real, levando em consideração tanto as características de *hardware* quanto de *software*. Este *profile* provê suporte para os estágios de especificação, criação e verificação/validação de sistemas. Ele foi criado para substituição do *UML Profile for Schedulability, Performance and Time* (OMG., 2011). A criação do MARTE foi possível através do mecanismo de *profile* da UML, que permite a extensão da linguagem. Este mecanismo permite criar novos elementos, de um domínio específico, sem que haja a necessidade da criação de uma nova linguagem de modelagem (OMG., 2011). Ademais, o MARTE é compatível com outros *profiles* da UML, incluindo o SysML.

MARTE tem por objetivo facilitar a modelagem de sistemas que necessitam de uma análise específica, e utiliza anotações para isto. Essas anotações são feitas por meio de *estereótipos*, sendo possível atribuir valores às propriedades necessárias, para realização de análise e obtenção de métricas desejadas. A especificação do MARTE é dividida em um conjunto de unidades de extensão que contêm anotações para especificar as características dos sistemas. Este trabalho utiliza a unidade de extensão para Modelagem de Recursos Genéricos (GRM), que tem por objetivo oferecer conceitos necessários para modelagem geral de plataformas para execução de aplicações embarcadas de tempo-real. Neste trabalho, é utilizado o *estereótipo ResourceUsage*, presente na GRM, e os valores marcados *execTime* e *energy*. O estereótipo descreve uma ação, enquanto o valor marcado representa o nome da propriedade, bem como o valor atribuído a ela. O valor marcado *execTime* indica o tempo de execução de uma tarefa, por exemplo *{execTime = (2,s)}*, indica a duração de 2 segundos para realização de uma tarefa. Já o valor marcado *energy* representa o consumo de energia para realização de uma tarefa, por exemplo *{energy = (155, j)}*, indica que uma tarefa consome 155 *joules* para ser completada. A documentação oficial do MARTE e mais informações sobre este *profile* podem ser encontradas em (OMG., 2011).

2.5 Redes de Petri

Rede de Petri é uma técnica de modelagem e análise de sistemas complexos muito difundida. Ela possui uma representação matemática e mecanismos de análise que permitem a verificação de características de sistemas. Seu conceito surgiu na tese de doutorado proposta por Carl Adam Petri em 1962 (PETRI, 1962). Desde então, tem sido utilizada para modelagem e análise de sistemas de várias áreas do conhecimento. O modelo mais básico das redes de Petri é conhecido como *Places/Transition*, contudo, a necessidade de atender a outras áreas do conhecimento fez com que outras extensões deste modelo foram surgissem ao longo do tempo, como as redes de Petri temporizadas (MERLIN; FARBER, 1976), estocásticas (MARSAN, 1990), alto-nível (JENSEN, 1991) e orientada a objetos (JANOUSEK, 1998). Algumas das

extensões das redes de Petri são mostradas na Figura 2.8.

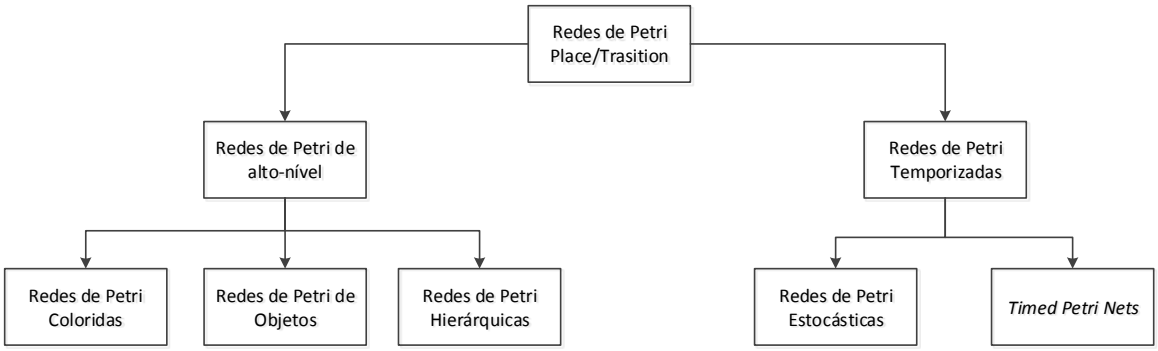


Figura 2.8: Tipos de redes de Petri.

Adaptado: OLIVEIRA et al. (2013).

Um exemplo de uma rede de Petri *Place/Transition* é mostrado na Figura 2.9. Os seguintes elementos compõem esse tipo de rede de Petri (MURATA, 1989):

- **Token.** Representa o recurso que aponta o estado em que o sistema se encontra em determinado momento.
- **Lugar.** Tem como principal função o armazenamento de *tokens*, indicando o estado o qual o sistema se encontra. Graficamente, os lugares são representados por círculos.
- **Transição.** Representa as ações realizadas pelo sistema, e portanto causam a mudança de estado do sistema. Graficamente, são representadas por barras retangulares.
- **Arco.** Representa o fluxo dos *tokens* pela rede. Um arco é dito *arco de entrada* quando tem origem em um lugar e destino uma transição, já quando tem uma transição como origem e um lugar como destino é chamado de *arco de saída* (OLIVEIRA, 2014). O arco também pode ter um peso associado, indicado por um número próximo ao arco (quando não existir essa notação, o peso é igual a 1), que reflete na quantidade de *tokens* gerados ou consumidos. Caso seja um arco de entrada, irá ser **consumida** a quantidade de *tokens* referente ao peso do arco, porém, se for um arco de saída, irá ser **gerada** a quantidade de *tokens* referente ao peso do arco.

Uma rede de Petri não permite a ligação por um arco entre elementos do mesmo tipo, ou seja, um lugar ligado a um lugar, ou uma transição ligada a outra transição. Além disso, para ser disparada, uma transição t precisa estar habilitada. Uma transição t está habilitada quando os lugares conectados a ela por um arco de entrada possuem uma quantidade de *tokens* igual ou superior ao peso destes arcos de entrada. O disparo da transição causa mudança na marcação da rede de Petri, o que indica uma mudança de estado do sistema ou de um elemento. O disparo faz

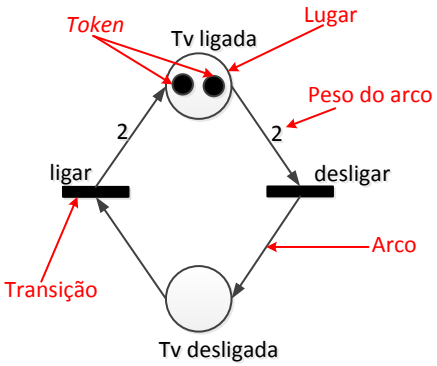


Figura 2.9: Exemplo de uma rede de Petri *Place/Transition*.

com que os *tokens* sejam consumidos, e novos *tokens* sejam gerados nos lugares conectados a transição por arcos de saída.

No exemplo da Figura 2.9, a presença dos *tokens* no lugar *Tv ligada* indica o estado atual do elemento, neste caso, que a tv está ligada. A transição *desligar* está habilitada, pois o arco de entrada possui peso 2, o mesmo número de *tokens* presente no lugar *Tv ligada*. Quando é disparada, a transição *desligar* consome os *tokens* presentes no lugar *Tv ligada* e gera 1 *token* no lugar *Tv desligada*. A partir deste momento, a transição *ligar* está habilitada, e quando é disparada consome o *token* do lugar *Tv desligada*, e gera 2 *tokens* para o lugar *Tv ligada*, pois o peso do arco de saída é igual a 2.

Formalmente, uma rede de Petri é definida da seguinte forma:

Definição 2.5.1. Uma rede de Petri é uma quíntupla, $R = (P, T, F, W, M_0)$, onde:

- $P = \{p_1, p_2, ..., p_n\}$, é um conjunto finito não-vazio de lugares;
- $T = \{t_1, t_2, ..., t_n\}$, é conjunto finito não-vazio de transições, $P \cap T = \{\emptyset\}$;
- $F \subseteq (P \times T) \cup (T \times P)$, é um conjunto de arcos;
- $W : F \longrightarrow IR^+ \cup \{0\}$, é a função de atribuição de peso aos arcos;
- $M_0 : P \longrightarrow IN$, é a função de marcação inicial, onde $P \cap T = \{\emptyset\}$ e $P \cup T \neq \{\emptyset\}$.

2.5.1 Propriedades das Redes de Petri

O estudo das propriedades de redes de Petri permite a análise do sistema modelado. Os tipos de propriedades podem ser divididos em duas categorias: as propriedades dependentes de marcação inicial, conhecidas como propriedades comportamentais, e as propriedades não dependentes de marcação, conhecidas como propriedades estruturais (MACIEL; LINS; CUNHA, 1996; MURATA, 1989).

2.5.1.1 Propriedade Comportamentais

As propriedades comportamentais são aquelas que dependem apenas da marcação inicial da rede de Petri. As propriedades apresentadas são alcançabilidade, limitação, segurança, *liveness* e cobertura.

Alcançabilidade ou *reachability* indica a possibilidade de uma determinada marcação ser atingida pelo disparo de um número finito de transições a partir de uma marcação inicial. Dada uma rede de Petri marcada $RM = (R, M_0)$, o disparo de uma transição t_0 altera a marcação da rede de Petri. Uma marcação M' é acessível a partir de M_0 se existe uma sequência de transições que, disparadas, levam à marcação M_0. Ou seja, se a marcação M_0 habilita a transição t_0, disparando-se essa transição, atinge-se a marcação M_1. A marcação M_1 habilita t_1 a qual, sendo disparada, atinge-se a marcação M_2 e assim por diante até a obtenção da marcação M'.

Seja um lugar $p_i \in P$, de uma rede de Petri marcada $RM = (R, M_0)$, esse lugar é k-limitado $(k \in IN)$ ou simplesmente limitado se para toda marcação acessível $M \in CA(R, M_0)$, $M(p_i) \leq k$.

O limite k é o número máximo de *tokens* que um lugar pode acumular. Uma rede de Petri marcada $RM = (R, M_0)$ é k-limitada se o número de *tokens* de cada lugar de RM não exceder k em qualquer marcação acessível de $RM(max(M(p)) = k, \forall p \in P)$.

Segurança ou *safeness* é uma particularização da propriedade de limitação. O conceito de limitação define que um lugar p_i é k-limitado se o número de *tokens* que esse lugar pode acumular estiver limitado ao número k. Um lugar que é 1-limitado pode ser simplesmente chamado de seguro.

Vivacidade ou *liveness* está definida em função das possibilidades de disparo das transições. Uma rede de Petri é considerada *live* se, independente das marcações que sejam alcançáveis a partir de M_0, for sempre possível disparar qualquer transição da rede de Petri através de uma sequência de transições $L(M_0)$. A ausência de bloqueio (*deadlock*) em sistemas está fortemente ligada ao conceito de vivacidade pois, *deadlock* em uma rede de Petri é a impossibilidade do disparo de qualquer transição da rede de Petri. O fato de um sistema ser livre de *deadlock* não significa que seja *live*, entretanto um sistema *live* implica em um sistema livre de *deadlocks*. O conceito de cobertura está associado ao conceito de alcançabilidade e de live. Uma marcação M_i é coberta se existir uma marcação $M_j \neq M_i$, tal que $M_j \geq M_i$.

2.5.1.2 Propriedade Estruturais

As propriedades estruturais são aquelas que dependem apenas da estrutura da rede de Petri. Essas propriedades refletem características independentes de marcação.

Uma rede de Petri $R = (P, T, F, W, M_0)$ é classificada como **estruturalmente limitada** se for limitada para qualquer marcação inicial.

Ela será considerada **consistente** se, disparando uma sequência de transições habilitadas a partir de uma marcação M_0 , retornar a M_0, porém todas as transições da rede de Petri são

disparadas pelo menos uma vez. Seja $RM = (R;M0)$ uma rede de Petri marcada e s uma sequência de transições, RM é consistente se $M_0[s > M_0]$ e toda transição T_i, disparar pelo menos uma vez em s.

2.5.2 Extensões Temporizadas de Redes de Petri

O conceito de tempo não foi definido originalmente para as redes de Petri *Place/Transition*. No entanto, para realização de análises em muitas aplicações, se faz necessário a utilização do tempo. Sem uma notação para o tempo, não é possível a realização de análises de desempenho (AALST; HEE; REIJERS, 2000). Nestes tipos de redes de Petri, o tempo pode estar relacionado a (MACIEL; LINS; CUNHA, 1996):

- **Lugares.** Onde *tokens* podem ter de permanecer naquele lugar por um tempo determinado.
- **Tokens.** Onde estes possuem uma informação que indica se pode ser consumido ou não.
- **Transições.** Quando habilitadas devem respeitar esse tempo como um atraso para o disparo.

Entretanto, na maioria das extensões o tempo é atribuído às transições. Quanto ao tipo de tempo que é associado às transições, eles podem ser (AALST; HEE; REIJERS, 2000):

- **Determinísticos.** Indicam o tempo exato de *delay* da transição, o que em termos de aplicação condiz com duração de uma atividade/evento;
- **Não-determinísticos.** Geralmente utilizam um intervalo para especificar a duração do *delay*;
- **Estocásticos.** O tempo é descrito através de distribuições de probabilidade (ex., normal, exponencial, F).

Outro conceito associado a atribuição de tempo às redes de Petri, é o grau de habilitação. Ele indica o número de vezes que uma transição pode ser disparada. Estas semânticas de disparo podem ser do tipo (ANDRADE, 2014):

- **Single-Server.** Apenas um *token* é disparado por vez, ou seja, a capacidade de um lugar/transição é 1.
- **Multiple-Server.** É possível fazer k disparos por vez, ou seja, a capacidade de um lugar/transição é um k inteiro.
- **Infinite-Server.** É possível fazer infinitos disparos de uma única vez.

Ademais, outro problema surge quanto se trabalha com transições temporizadas, o conflito entre transições. Quando a habilitação e disparo de uma transição causa a desabilitação de uma outra transição, existe a situação de conflito. Para resolver este problema, existem três tipos de abordagens (AALST; HEE; REIJERS, 2000):

- **Age Memory.** O tempo que resta para a transição ser habilitada é congelado no momento que a transição torna-se desabilitada, e é retomado no momento em que a transição se torna habilitada novamente;
- **Enabling Memory.** Após cada disparo, o tempo de disparo de transições que ficaram desabilitadas tem de começar de zero;
- **Reset Memory.** Após cada disparo, um novo tempo de disparo é atribuído a transição.

2.5.2.1 Redes de Petri Determinísticas e Estocásticas

A Rede de Petri Estocástica e Determinística (DSPN) (MARSAN; CHIOLA, 1987) (GERMAN, 2000) é uma extensão das redes de Petri *Place/Transition* onde existem transições temporizadas (determinísticas e estocásticas) e transições imediatas. Os tempos de disparado das transições estocásticas são exponencialmente distribuídos, por esse motivo, são também chamadas de transições exponenciais. Apenas transições imediatas e exponenciais são usadas neste trabalho, porém, transições determinísticas podem ser usadas em trabalhos futuros.

Formalmente as DSPNs são definidas da seguinte forma (GERMAN, 2000):

Definição 2.5.2. Uma DSPN é tupla, $R = (P, T, I, O, H, \Pi, G, M_0, Atts)$, onde:

- $P = \{p_1, p_2, ..., p_n\}$, é um conjunto finito não-vazio de lugares;
- $T = \{t_1, t_2, ..., t_n\}$, é conjunto finito não-vazio de transições;
- $I \in (\mathbb{N}^n \longrightarrow \mathbb{N})^{n \times m}$, é a matriz que representa os arcos de entrada (podem ser dependentes de marcação);
- $O \in (\mathbb{N}^n \longrightarrow \mathbb{N})^{n \times m}$, é a matriz que representa os arcos de saída (podem ser dependentes de marcação);
- $H \in (\mathbb{N}^n \longrightarrow \mathbb{N})^{n \times m}$, é a matriz que representa os arcos inibidores (podem ser dependentes de marcação);
- $\Pi \in \mathbb{N}^m$, é o vetor que associa o nível de prioridade para cada transição;
- $G \in (\mathbb{N}^n \longrightarrow \{true, false\})^m$, é o vetor que associa uma função de guarda para cada transição que pode depender da marcação atual. Uma função de guarda atribui uma função booleana a uma transição da rede de Petri. O resultado de uma função de guarda controla

o disparo de uma transição. Por exemplo, se uma função de guarda definida por $\#P_0 > 1$ é atribuída a uma transição t_n, indica que a transição t_n só será disparada caso o número de *tokens* no lugar P_0 for maior que um;

- $M_0 \in \mathbb{N}^n$, é o vetor que define a marcação inicial;
- $Atts = (Dist, W, Markdep, Policy, Concurrency)^m$, é definido pelo conjunto de atributos:

  i. $Dist \in \mathbb{N}^n \longrightarrow \mathscr{F}$, é uma função de distribuição de probabilidade associada ao tempo de uma transição, sendo que o domínio de $\mathscr{F}$ é [0, ∞] (a distribuição pode ser dependente da marcação);

  ii. $W \in \mathbb{N}^m \longrightarrow \mathbb{R}^+$, é a função peso, que associa um peso (w_t) às transições imediatas e uma taxa λ_t às transições temporizadas, onde:

$$W(t) = \begin{cases} w_t \geq 0, & \text{se } t \text{ é uma transição imediata,} \\ \lambda_t > 0, & \text{caso contrário.} \end{cases}$$

  iii. $Markdep \in \{constant, enabdep\}$, define se a distribuição de probabilidade associada ao tempo de uma transição é constante (*constant*) ou dependente de marcação (*enabdep*);

  iv. $Policy \in \{prd, prs\}$, é a política de preempção adotada pela transição (*prd - preemptive repeat different* - corresponde à *reset memory* e *prs - preemptive resume* - possui significado idêntico à *age memory policy*);

  v. $Concurrency \in \{ss, is\}$ é o grau de concorrência das transições, onde *ss* representa a semântica de *Single-Server* e *is* significa a semântica de *Infinite-Server* (mesmo sentido dos modelos de fila).

Para mais informações sobre redes de Petri e suas extensões, os leitores podem se referir a (MURATA, 1989), (MARSAN, 1990), (MACIEL; LINS; CUNHA, 1996), (AALST; HEE; REIJERS, 2000), (MARSAN; CHIOLA, 1987), (GERMAN, 2000), (JENSEN, 1995) e (SIFAKIS, 1977).

2.6 Diagrama de blocos de Confiabilidade

Diagrama de Blocos de Confiabilidade (*Reliability Block Diagram* - RBD) é um tipo de modelo combinatório proposto inicialmente para analisar a confiabilidade de sistemas baseados nas relações de seus componentes. Posteriormente, este formalismo foi estendido para a análise de disponibilidade e manutenibilidade.

Um RBD é formado pelos seguintes componentes: vértice de origem, vértice de destino, componentes do sistema (representados por blocos) e arcos conectando os blocos e os vértices.

Um sistema modelado em RBD está disponível caso haja algum caminho do vértice de origem até o vértice de destino, no qual componentes indisponíveis representam uma interrupção em um segmento do caminho. Na Figura 2.10 (a) ilustramos este fato, onde é mostrado que o sistema permanece disponível mesmo com a falha de dois componentes, uma vez que há um caminho contínuo dos vértices de início e fim do RBD. Já na representação do sistema indisponível (Figura 2.10 (b)), a falha do componente é crítica para o sistema, uma vez que ele impede que haja um caminho contínuo do vértice de origem até o vértice de destino.

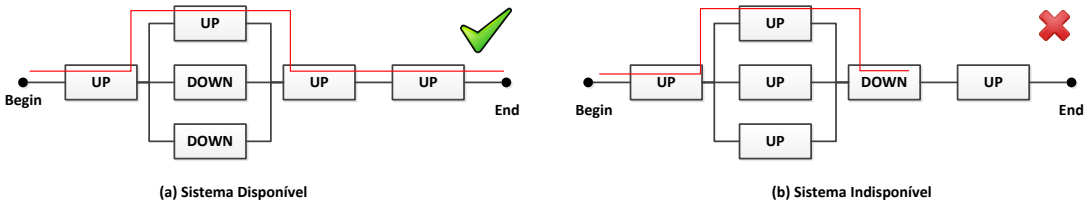


Figura 2.10: Exemplo de Disponibilidade em RBD.

Há duas formas principais de arranjar os blocos, como ilustra a Figura 2.11: em série e em paralelo. Em um conjunto de componentes em série, todos os componentes deverão estar operacionais para que conjunto esteja disponível. Em conjunto de componentes em paralelo, por sua vez, é necessário que pelo menos um esteja disponível. Outra estrutura típica é a *k-out-of-n*, que é formada por n componentes, e necessita do funcionamento de pelo menos k componentes para estar operacional.

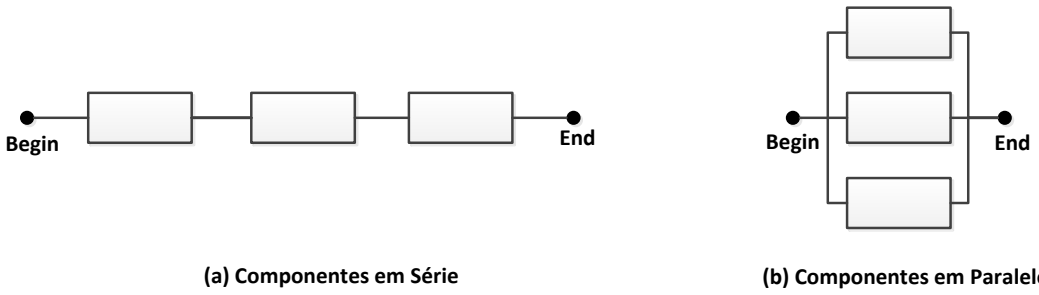


Figura 2.11: Tipos de agrupamentos de componentes em um RBD.

Para avaliar a disponibilidade através de um RBD, introduzimos antes o conceito de função estrutural de um sistema. Considere que um sistema S é formado por n componentes: $c_1, c_2, c_3, \ldots, c_n$, e este sistema pode estar em falha ou operacional, de acordo com o estado de seus componentes. O estado de um componente c_i é dado pela variável aleatória x_i, tal que:

$$x = \begin{cases} 0, & \text{se o componente falhou} \\ 1, & \text{se o componente está operacional} \end{cases}$$

A confiabilidade de dois blocos conectados em série (ver Figura 2.11 (a)) pode ser obtida

através da Equação 2.1 a seguir:

$$R_s(t) = R_1(t) \times R_2(t) \tag{2.1}$$

onde, $R_1(t)$ é a confiabilidade do bloco 1, e $R_2(t)$ é a confiabilidade do bloco 2.

De uma forma geral, considerando um sistema com n elementos, a confiabilidade do modelo em série pode ser obtida através da seguinte Equação 2.2:

$$R_s(t) = \prod_{i=1}^{n} R_i(t) \tag{2.2}$$

onde, $R_i(t)$ é a confiabilidade do bloco R_i.

A confiabilidade de dois blocos conectados em paralelo pode ser obtida através da seguinte Equação 2.3:

$$R_p(t) = 1 - \prod_{i=1}^{2} (1 - R_i(t)) \tag{2.3}$$

Para esse tipo de sistemas temos que pelo menos um componente deve ser operacional para que todo sistema esteja funcionando. De uma forma geral, considerando n componentes, a confiabilidade do sistema pode ser obtida através da seguinte Equação 2.4:

$$R_p(t) = 1 - \prod_{i=1}^{n} (1 - R_i(t)) \tag{2.4}$$

onde, $R_i(t)$ é a confiabilidade do bloco b_i.

Modelos RBDs são utilizados em sistemas que contêm módulos independentes, onde cada um pode ser facilmente representado por um bloco de confiabilidade. Assim, havendo a necessidade de modelar sistemas complexos, onde se exige a adição de redundância em módulos do sistema, o usuário tem que recorrer a técnicas de modelagem hierárquica, fazendo uso, em conjunto, de modelos como redes de Petri e RBD, na tentativa de obter resultados mais expressivos.

2.7 Considerações Finais

Este capítulo apresentou os conceitos principais acerca deste trabalho. Primeiramente, uma introdução a conceitos sobre avaliação de desempenho e consumo de energia foi apresentado. Depois, foram abordados os fundamentos sobre computação em nuvem e *Mobile Cloud Computing* (MCC). Posteriormente, foram explanados os conceitos sobre a *System Modelling Language* (SysML) e os diagramas que são utilizados na pesquisa, além do *profile* MARTE. Também foram apresentados conceitos introdutórios acerca das redes Petri, mostrando que elas são uma ferramenta de grande valor para a modelagem e análise dos mais variados tipos de sistemas. Por fim, foram apresentados conceitos acerca do Diagrama de blocos de Confiabilidade.

3

Mapeamento dos Diagramas da SysML em DSPNs

A guerra se baseia no engano, se faz pelo ganho e se adapta pela divisão e combinação.

—SUN TZU, A ARTE DA GUERRA

Este capítulo descreve a abordagem proposta para realização do mapeamento de diagramas da SysML em DSPNs. Após a obtenção das DSPNs, análises quantitativas podem ser realizadas, a fim de observar métricas de interesse. A abordagem apresentada pode auxiliar projetistas e desenvolvedores de aplicações móveis que possuem pouca experiência em modelagem estocástica a analisar métricas ligadas a desempenho, consumo de energia e disponibilidade. Além disso, estudos de diferentes cenários e formas de desenvolvimento de uma aplicação móvel poderão ser realizados de maneira mais rápida, com menor custo e menos consumo de recursos. O processo de mapeamento é apresentado desde a modelagem de alto nível até a obtenção dos modelos analíticos, os quais são adotados para análises.

3.1 Abordagem adotada

Na Figura 3.1 é mostrada a abordagem utilizada neste trabalho para análise de métricas de desempenho, consumo de energia e disponibilidade. Essa abordagem foi baseada na proposta do *framework* apresentado por ANDRADE (2014). Contudo, foram realizadas algumas adaptações para se trabalhar com métricas relacionadas ao desempenho e consumo de energia de aplicações móveis. Não foram consideradas operações modeladas a partir do Diagrama de Atividades da SysML (SysML-AD). O mapeamento é feito principalmente a partir do Diagrama de Estados da SysML (SysML-STM). Também foram adotados diferentes estereótipos e valores marcados do *profile* MARTE para trabalhar com métricas de desempenho e consumo de energia. Além disso, não foi utilizada a abordagem hierárquica com RBD para cálculo da disponibilidade do sistema.

Para isso, foi utilizada as análises com DSPNs.

A abordagem é composta por quatro passos: (i) Definir cenário e funcionalidades a serem avaliados, ou seja, é definido o escopo que a análise visa abranger; (ii) Modelar os cenários usando os diagramas da SysML e as anotações de MARTE; (iii) Mapear os modelos da SysML em DSPNs. Nessa fase, se necessário, o projetista deverá atribuir as funções de guarda às DSPNs obtidas; E por fim, (iv) calcular métricas relacionadas ao desempenho, consumo de energia e disponibilidade, através de análises nas DSPNs obtidas.

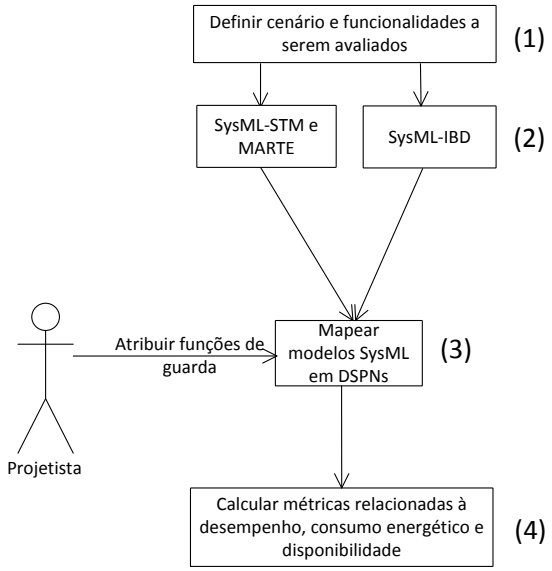


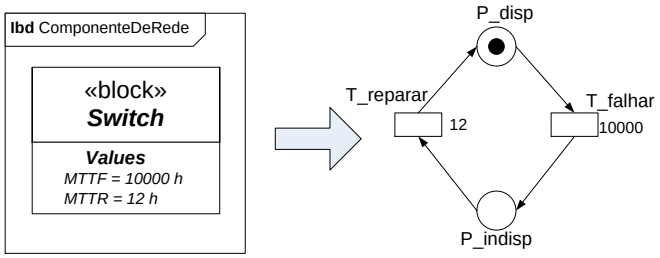
Figura 3.1: Abordagem adotada neste trabalho

No primeiro passo (1) da abordagem é definido o escopo do projeto a ser estudado, definindo quais componentes do cenário que se deseja estudar serão levados em consideração. Nesse passo projetistas definem a infraestrutura que a aplicação móvel utilizará. Por exemplo, a quantidade de servidores disponíveis para oferecer o serviço à aplicação. No segundo passo (2) são criados os diagramas de bloco interno e de estados (SysML-IBD e SysML-STM, repectivamente). O diagrama de bloco interno representa a visão estática do sistema e é importante para ter um apoio visual da relação entre os elementos presentes no cenário a ser estudado. Além disso, no SysML-IBD são apresentadas algumas propriedades dos elementos do cenário, tais como propriedades de Tempo Médio até a falha (MTTF), Tempo Médio de reparo (MTTR) e anotações de alocação. Na criação do SysML-STM é levado em conta o comportamento desejado da aplicação em termos de estado, por exemplo, quais ações serão executadas e quais os estados da aplicação, além de apresentar o tempo de execução e consumo médio de energia de cada ação. No terceiro passo (3) é feito o mapeamento dos diagramas SysML em DSPNs. O mapeamento consiste em seguir regras (apresentadas nas próximas seções) para se obter modelos DSPNs através de diagramas da SysML. Ao se obter os modelos DSPNs, é função do projetista atribuir as funções de guarda às transições. Por último, no quarto passo (4), são realizadas análises

nas DSPNs para se obter as estimativas das métricas de desempenho (tempo de execução), consumo de energia e disponibilidade. As análises são realizadas através de software que suporte modelagem e análises de redes de Petri, tais como TimeNET (ZIMMERMANN; KNOKE, 2007) e Mercury (SILVA, 2013). Nesse passo também é função do projetista escrever a expressão lógica que resultará na estimativa das métricas, já que, até agora, o mapeamento é feito de forma manual. Posteriormente, o projetista pode tomar as decisões de implementação da aplicação móvel com base nas análises numéricas das métricas que lhe são apresentadas.

3.2 Mapeamento do SysML-IBD

O mapeamento do SysML-IBD pode ser feito de duas formas. Na primeira forma de mapeamento, quando o bloco possui em seus compartimentos de propriedades os valores de MTTF e MTTR, a DSPN correspondente do bloco é dada por dois lugares e duas transições exponenciais interligados (ver Figura 3.2). Os valores do MTTF e MTTR assinados no bloco são atribuídos às transições exponenciais como *delays*, fazendo com que a DSPN, nesse caso, represente o comportamento de falha-reparo do elemento que é representado pelo bloco.

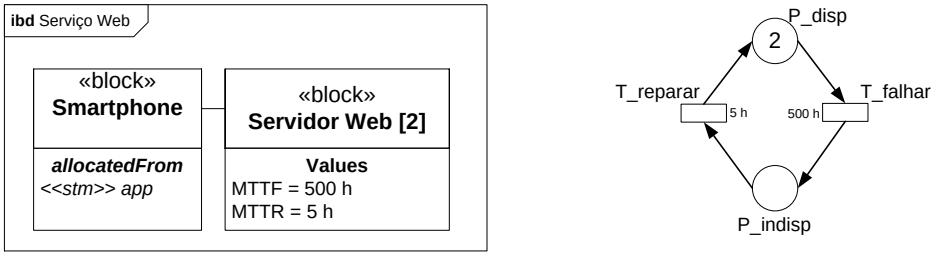


(a) Exemplo de SysML-IBD **(b)** DSPN correspondente ao bloco

Figura 3.2: Exemplo de mapeamento de um bloco do SysML-IBD em uma DSPN.

Na DSPN obtida da Figura 3.2 (b), a operacionalidade do bloco *switch* é indicada pela presença de um *token* no lugar *P_disp*. Já a indisponibilidade do *switch* é indicada quando o *token* está presente no lugar *P_indisp*. A mudança do estado disponível para o estado indisponível, e vice-versa, é dada pelo disparo das transições exponenciais *T_falhar* e *T_reparar*. Quando o *switch* falha, o *token* presente no lugar *P_disp* é consumido no disparo da transição *T_falhar*, e ele então, é colocado no lugar *P_indisp*. Quando o *switch* é reparado, a transição *T_reparar* é disparada e o *token* volta ao lugar *P_disp*. Os tempos de disparo das transições *T_falhar* e *T_reparar* obedecem aos valores do MTTF e MTTR assinados na propriedade do bloco do SysML-IBD, respectivamente.

Ainda na primeira forma de mapeamento, quando o componente modelado através do SysML-IBD possuir redundância, então será indicado através de um número inteiro entre colchetes junto ao título do bloco (ver Figura 3.3 (a)). Quando tal elemento for mapeado para uma DSPN, o número de redundância do elemento, indicado entre os colchetes no SysML-IBD, será representado pela quantidade de *tokens* no lugar inicial da DSPN (ver Figura 3.3 (b)).



(a) SysML-IBD representando Smartphone e Servidor Web com redundância **(b)** DSPN correspondente ao bloco do Servidor Web

Figura 3.3: Exemplo de mapeamento de blocos com redundância.

A segunda forma de mapeamento do SysML-IBD(ii) ocorre quando o bloco contiver em um de seus compartimentos de propriedades a marcação de alocação ***allocatedFrom*** (ver Figura 3.3 (a)). Isso denota que o comportamento do bloco será descrito através do SysML-STM associado ao bloco. Portanto, a obtenção da DSPN correspondente se dará pelo mapeamento do SysML-STM. As regras para o mapeamento do SysML-STM são descritas a seguir.

3.3 Mapeamento do SysML-STM

O SysML-STM é composto por cinco elementos básicos: estado, estado inicial, estado final, transição, e escolha. Aplicando as regras de mapeamento nesses elementos básicos, é possível combiná-los e obter a DSPN correspondente de um SysML-STM. As regras que definem o mapeamento de cada um desses elementos são mostradas na Figura 3.4 e detalhadas a seguir.

O estado representa uma situação de um elemento do sistema em um determinado momento. No SysML-STM ele é representado por um retângulo com um identificador. O elemento que representa o estado na DSPN é um lugar, um círculo contendo um identificador (ver Figura 3.4 (a)).

Uma seta conectando dois estados representa uma transição/ação no SysML-STM. A transição é responsável pela mudança de estados do sistema. No SysML-STM, ela é representada por uma seta com um identificador. Seu elemento corresponde em uma DSPN é uma transição imediata com um identificador da ação (ver Figura 3.4 (b)). Entretanto, se a transição no modelo do SysML-STM estiver anotada com o estereótipo *ResourceUsage* e o valor marcado *execTime* do *profile* MARTE, a transição imediata é substituída por uma transição exponencial no modelo DSPN, e o valor atribuído ao *execTime* é passado como *delay* para a transição exponencial (Figura 3.4 (f)).

O estado inicial indica onde a execução do sistema modelado inicia. No SysML-STM, é obrigatório existir o estado inicial e somente é permitido um estado inicial no modelo. O estado inicial é representado por um círculo preto no SysML-STM. Seu elemento correspondente na DSPN é a adição de um *token* no lugar indicado pela seta do estado inicial no SysML-STM

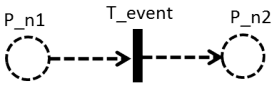
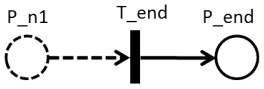
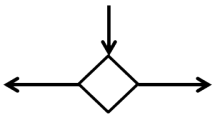
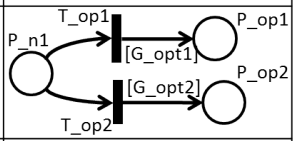
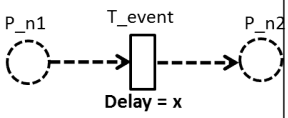
	SysML (+ MARTE)		DSPN	
a	**Estado**	*Nome do Estado*	Lugar	
b	**Transição/Evento**	*Nome da Transição*	Transição imediata conectada por dois arcos	
c	**Estado Inicial**		1 *token* no lugar	
d	**Estado final**		Transição imediata seguido por um lugar	
e	**Escolha**		Lugar seguido de duas transições imediatas, com funções de guarda e dois lugares de saída	
f	**Tempo de execução = *x***	*<<ResourceUsage>>* **Nome da transição** {*execTime* = exp(x) }	Transição exponencial , com *x* correspondendo ao *delay* da transição	
g	**Consumo de energia = *y***	*<<ResourceUsage>>* **Nome da transição** {*energy* = 'y'}	Métrica observada por análise do modelo	Energia estimada = $T(t_j) * E_{am}(f) * t$

Figura 3.4: Regras de mapeamento dos elementos do SysML-STM em DSPNs.

(Figura 3.4 (c)). Já o estado final é responsável por determinar o fim da execução do sistema modelado. A presença do estado final é opcional no modelo e também pode existir mais de um estado final no mesmo SysML-STM. o estado final é representado por pequeno círculo preto cercado por um círculo branco, no SysML-STM. Para representar o estado final no modelo DSPN é usado uma transição imediata seguida de um lugar (Figura 3.4 (d)).

A estrutura de escolha é utilizada para determinar o fluxo de execução do sistema. No SysML-STM, a estrutura de escolha apresenta uma possível saída de uma decisão. Ela é representada no SysML-STM por um losango com uma seta indicando a entrada e setas indicando as possíveis saídas. No modelo DSPN, a estrutura de escolha é mapeada em um lugar seguido de duas transições imediatas com dois lugares de saída diferentes (ver Figura 3.4 (e)). A estrutura deve satisfazer uma condição *bolleana*, quando uma condição é satisfeita, a transição imediata correspondente é disparada. Em Redes de Petri essa condição é conhecida como *função de guarda*.

Por fim, quando uma transição do SysML-STM é anotada com o estereótipo *ResourceUsage* e o valor marcado *energy* de MARTE (Figura 3.4 (g)), indica o consumo médio de energia daquela ação. Note que para obter o valor estimado do consumo de energia do modelo DSPN

obtido pelo mapeamento deve-se utilizar da equação (3.1) mostrada abaixo:

$$E_e(f) = T(t_j) * E_{am}(f) * t \tag{3.1}$$

Onde, $E_e(f)$ corresponde ao consumo de energia estimado da funcionalidade f; $T(t_j)$ representa o *throughput* da transição t_j; $E_{am}(f)$ corresponde a média da energia medida da funcionalidade f; t corresponde ao tempo de observação. O *throughput* de uma transição é a vazão que aquela transição apresenta em relação ao sistema. Formalmente, ele definido por:

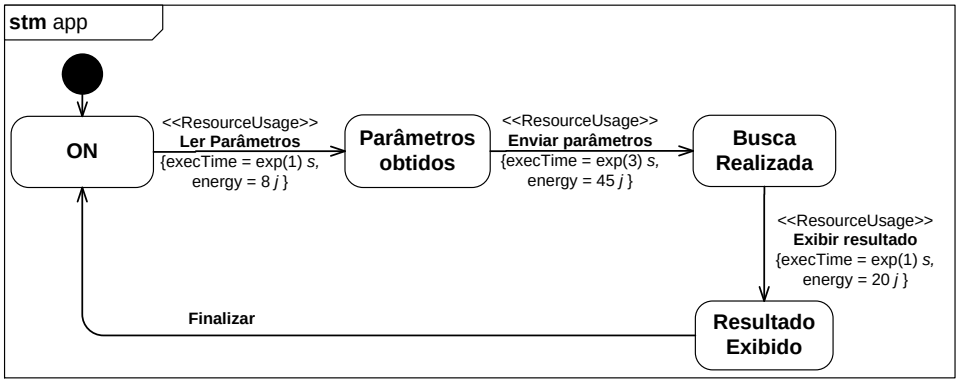
$$T(tj) = p_i * \lambda(t_j, W) * q_{ij} \tag{3.2}$$

onde,

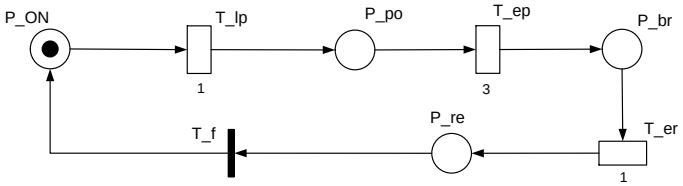
- $i \in S_2$ e $S_2 = \{i \in \{1,2,...S)\} : M_i[t_j >\}$;
- p_i é a probabilidade estacionária de uma marcação M_i que habilita t_j;
- $\lambda(t_j, W)$ é a taxa associada a transição;
- $$q_{ij} = \begin{cases} 1, & \text{se } \#O(I(t_j)) = 1 \\ x, & \text{caso contrário.} \end{cases}$$

A Figura 3.5 (a) apresenta um exemplo do comportamento de uma aplicação presente no *smartphone* representado no bloco da Figura 3.3 (a) através do SysML-STM. A aplicação colhe parâmetros fornecidos pelo usuário, e os envia para realização de uma busca nos servidores, quando o resultado da busca é recebido, o resultado é exibido. A aplicação inicia no estado *ON*, onde aguarda por entradas do usuário para colher os parâmetros de busca. Quando é feita a leitura dos parâmetros, através da ação *Ler Parâmetros*, a aplicação passa para o estado *Parâmetros obtidos*. Note que nesse estado os parâmetros da requisição foram obtidos para serem enviados aos servidores. Após ler os parâmetros, estes são enviados aos servidores para realização de uma busca, através da ação *Enviar parâmetros*. Quando o resultado da busca é recebido, a aplicação alcança o estado *Busca realizada*. Depois, a ação *Exibir resultado* exibe os resultados da busca realizada anteriormente, e a aplicação atinge o estado *Resultado exibido*. Por fim, a ação *Finalizar* é executada, a aplicação volta ao estado *ON*. Ainda na Figura 3.5 (a) é possível observar o uso das anotações de MARTE. Foi utilizado o estereótipo *ResourceUsage*, e os valor marcados *execTime* e *energy* para representar o tempo de execução e consumo médio de energia de cada ação. Por exemplo, a ação *Enviar parâmetros* é executada em um tempo exponencial de 3 segundos e tem um consumo médio de energia de 45 *joules*.

A DSPN obtida através do mapeamento do SysML-STM, seguindo as regras apresentadas anteriormente, é mostrada na Figura 3.5 (b). Cada estado do SysML-STM é mapeado em um lugar na DSPN. As transições do SysML-STM anotadas com o estereótipo *ResourceUsage* e o



(a) SysML-STM da aplicação presente no *smartphone*



(b) DSPN correspondente ao SysML-STM

Figura 3.5: Exemplo de mapeamento de um SysML-STM.

valor marcado *execTime* do *profile* MARTE são mapeadas em transições exponenciais no modelo DSPN, sendo associados como *delays* em suas respectivas transições exponenciais. Os valores associados às anotações de MARTE do valor marcado *energy*, são utilizados durante análises na DSPN para obter o consumo estimado de energia. Note também que a transição sem anotação de MARTE é mapeada em uma transição imediata na DSPN. Para se estimar o consumo médio de energia durante as análises na DSPN, deve ser escrito a expressão lógica da métrica no software de análise da DSPN que segue a equação 3.1 apresentada anteriormente. Por exemplo, para estimar o consumo de energia da atividade *Enviar Parâmetros* (ver Figura 3.5), deve-se utilizar a equação 3.3.

$$E_e(EnviarParametros) = (P\{\#P\_ON > 0\}) * \frac{1}{3} * 45 * 4.8 \tag{3.3}$$

A primeira parte da equação $(P\{\#P\_ON > 0\}) * \frac{1}{3}$, representa o cálculo do *throughput* da transição *T_lp*. Separadamente, a função $(P\{\#P\_ON > 0\})$ representa a probabilidade estacionária do lugar *#P_ON* conter *tokens*, ou seja, ser maior que zero (> 0). O *45* representa a $E_{am}(EnviarParametros)$ e o *4.8*, o *tempo de observação*. O valor do tempo médio de execução do modelo é utilizado como tempo de observação na equação. O valor do tempo médio de execução pode ser calculado através de análises no modelo DSPN. Mais adiante é explicado como ele é obtido.

Para obter o valor total da energia estimada de toda a aplicação, é feito a soma da energia estimada das atividades/funcionalidades. O procedimento é mostrado na equação 3.4.

$$E_e(app) = ((P\{\#P\_ON > 0\} * \frac{1}{1}) * 8 * 4.8) + ((P\{\#P\_po > 0\} * \frac{1}{3}) * 45 * 4.8) + ((P\{\#P\_br > 0\} * \frac{1}{1}) * 20 * 4.8) \quad (3.4)$$

Por fim, para visualização do valor da métrica, a equação 3.4 é inserida no software de análises (neste trabalho foi utilizado o TimeNET (ZIMMERMANN; KNOKE, 2007)), que apresentará o resultado.

Para estimar o tempo médio de execução da DSPN é utilizada a equação que calcula o tempo de absorção do modelo. O *tempo de absorção* representa o tempo médio em que o modelo é executado por completo. Essa equação é representada pelo inverso do *throughput* da primeira transição do sistema, observando a ordem de disparo das transições a partir do estado inicial da DSPN. No exemplo da DSPN da Figura 3.5 (b), a primeira transição é a transição *T_lp*, pois é a primeira transição a ser disparada após o início da execução da DSPN a partir do estado inicial *P_ON*. Dessa forma, o cálculo do tempo de absorção do modelo é dada pela equação 3.5:

$$T_{ab} = \frac{1}{T(tj)} \quad (3.5)$$

onde, T_{ab} é o tempo de absorção obtido e $T(tj)$ é o *throughput* da primeira transição do modelo.

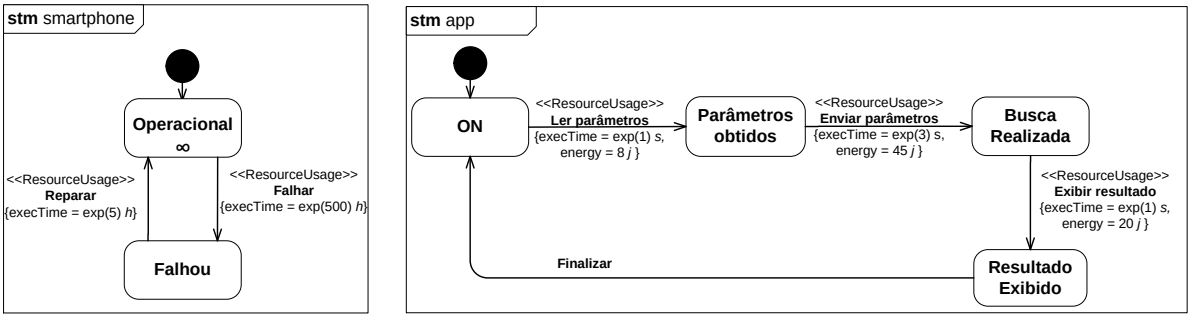
Sendo assim, para estimar o tempo médio de execução da DSPN da Figura 3.5 (b), o resultado seria obtido através da equação 3.6.

$$T_{ab} = \frac{1}{(P\{\#P\_ON > 0\}) * \frac{1}{1}} \quad (3.6)$$

Quanto a execução da DSPN (Figura 3.5 (b)), o *token* indica o recurso da aplicação, mostrando em qual estado ela se encontra. Ela inicia no lugar *P_ON*. Quando a transição *T_lp* é disparada, indica a leitura dos parâmetros fornecidos pelo usuário à aplicação. O *token* então é consumido pela transição e outro é gerado no lugar *P_po*, indicado que os parâmetros já foram obtidos pela aplicação. Após isto, a transição *T_ep* está habilitada. Quando ela é disparada, o *token* é consumido pela transição, indicando o envio dos parâmetros para os servidores, para que a busca seja realizada. Quando um *token* é depositado no lugar *P_br*, indica que a busca foi realizada e os resultados recebidos. Agora, a transição *T_er* está habilitada. Quando ela é disparada, os resultados da busca são exibidos na aplicação. O *token* é consumido pela transição e um novo *token* é gerado no lugar *P_re*, indicando que o resultado foi exibido pela aplicação. Por fim, a transição imediata *T_f* é disparada, indicando o fim da execução do serviço. Um *token* é depositado no lugar *P_ON* e aplicação está novamente no estado *ON*.

3.3.1 Mapeamento de Estados Compostos

Estados compostos são estados que contêm dois ou mais estados aninhados, ou seja, estados que possuem subestados. Eles são utilizados para simplificar e auxiliar a modelagem de sistemas complexos. Visualmente, no diagrama de estados da SysML eles se diferenciam dos demais por possuir o símbolo de infinito (∞) dentro deles. Na Figura 3.6 (a), é mostrado um exemplo de um SysML-STM que contém um estado composto. O estado *Operacional* é um estado composto (indicado pelo símbolo ∞), e o detalhamento do estado composto é definido pelo diagrama **stm app** (Figura 3.6 (b)).



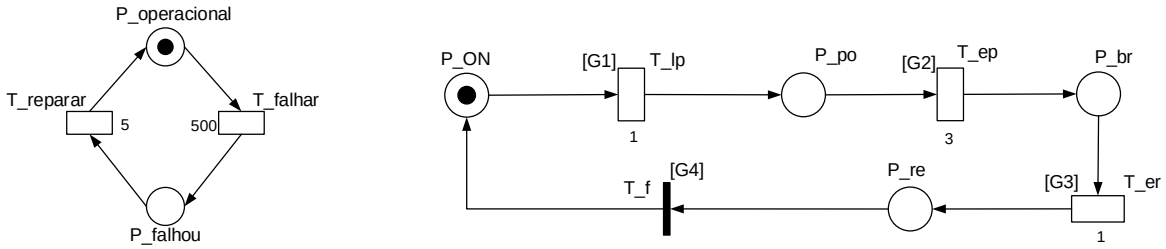
(a) SysML-STM contendo um estado composto

(b) SysML-STM que representa o comportamento do estado composto

Figura 3.6: Exemplo de um SysML-STM com um estado composto

As regras de mapeamento para os estados compostos são semelhantes às apresentadas anteriormente. Na Figura 3.7, são mostradas as DSPNs obtidas através do processo de mapeamento dos diagramas da Figura 3.6. Cada diagrama (**stm smartphone** e o **stm app**) é mapeado em uma DSPN (ver Figuras 3.7 (a) e (b), respectivamente), seguindo as regras apresentadas anteriormente. Porém, o mapeamento dos subestados do estado composto (estado *Operacional* da Figura 3.6) é levemente diferente. Os subestados (Figura 3.6 (b)) do estado composto só estarão operacionais enquanto o sistema permanece no estado *Operacional* do **stm smartphone**. Para lidar com essa restrição, são utilizadas funções de guarda (G1, G2, G3, e G4 - Figura 3.7 (b)), que verificam se o estado composto está ativo. No exemplo, as transições da DSPN da Figura 3.7 (b), só podem ser disparadas enquanto um *token* da DSPN da Figura 3.7 (a) permanece no lugar *P_operacional*, indicando que o *smartphone* está funcionando. Caso o *token* esteja no lugar *P_falhou*, indicando falha do *smartphone*, as transições da DSPN correspondente aos subestados do estado *Operacional* (Figura 3.7 (b)) não poderão ser disparadas. As funções de guarda utilizadas na DSPN da Figura 3.7 (b) são detalhadas na Tabela 3.1. Note que apesar das funções serem iguais, cada transição deve ter definida a sua *função de guarda*. A função de guarda de uma transição não pode ser atribuída a outra transição. Por exemplo, *G1* não pode estar atribuída a *T_lp* e *T_ep* ao mesmo tempo, no entanto, *G1* pode ser igual a *G2*. Uma limitação presente nesta abordagem, é que se o *smartphone* falhar e depois voltar a funcionar, a DSPN que representa o estado composto irá reiniciar a partir do estado onde foi interrompida. O que nem sempre pode representar o comportamento correto desejado para aplicação. Porém, se o

software for produzido utilizando essa mesma forma de funcionamento, a limitação é anulada.



(a) DSPN correspondente ao **stm smartphone** **(b)** DSPN correspondente ao **stm app**

Figura 3.7: Mapeamento dos SysML-STM da Figura 3.6.

Tabela 3.1: Funções de guarda utilizadas na DSPN da Figura 3.7 (b).

Guarda	**Função**
G1, G2, G3, G4	#P_operacional = 1

3.4 Considerações Finais

Este capítulo apresentou as regras de mapeamento propostas para avaliação de desempenho, consumo de energia e disponibilidade de aplicações para dispositivos móveis. O método baseia-se no mapeamento de modelos semiformais, descritos através dos diagramas da SysML, em DSPNs. Essa abordagem permite que projetistas, mesmo sem profundos conhecimentos em modelos analíticos, possam realizar análises de desempenho e consumo de energia em aplicações móveis ainda na fase inicial de seu desenvolvimento. Tal abordagem pode permitir tornar viável a criação de aplicações mais eficientes sem um alto custo associado ao processo de análise do projeto de software.

4

Modelos e Experimentos

Emancipate yourselves from mental slavery, none but ourselves can free our minds.

—BOB MARLEY

Neste capítulo é apresentado como os estudos foram conduzidos para demonstrar o uso da abordagem apresentada que tem por objetivo a realização de análises de desempenho, consumo de energia e disponibilidade de aplicações móveis. Inicialmente, um cenário base foi adotado para realização dos estudos e uma aplicação móvel foi desenvolvida para ser utilizada. Experimentos foram realizados utilizando este cenário base para obtenção de parâmetros de consumo de energia e tempo de execução, que serviram como *input* para os modelos gerados.

4.1 Visão Geral

Para condução dos estudos foi adotado um cenário base, mostrado na Figura 4.1. Ele é composto por um dispositivo móvel, um ponto de acesso WiFi, que fornece uma conexão WLAN, e um serviço hospedado em um servidor virtual que, por sua vez, está presente em um dos nós do *cluster* da MCC. O dispositivo móvel contém uma aplicação que consome o serviço provido pelo servidor web. A conexão WLAN, fornecida pelo ponto de acesso WiFi, é responsável pela comunicação entre o dispositivo móvel e a MCC.

A aplicação móvel criada, que executa no dispositivo móvel, tem como principal funcionalidade a realização do *upload* de arquivos para os servidores alocados na MCC. Seu funcionamento se dá da seguinte forma: primeiro, o usuário tira uma foto, a imagem capturada é então comprimida pela aplicação móvel, utilizando a classe *ZipEntry* do Java (ORACLE, 2014a). Por fim, é realizado o *upload* da imagem para os servidores alocados na MCC.

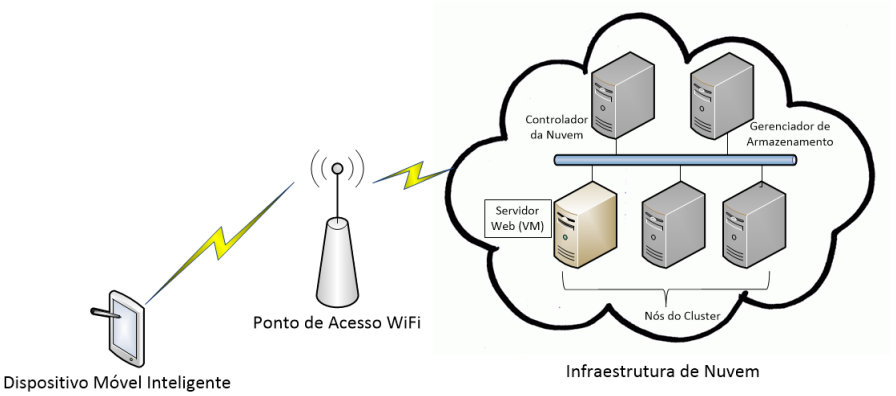


Figura 4.1: Cenário base adotado para estudo

4.2 Obtenção dos parâmetros de tempo de execução e consumo de energia

A infraestrutura real do cenário base, mostrada na Figura 4.1, foi montada em laboratório para que fosse possível obter os parâmetros de tempo de execução e consumo de energia da aplicação móvel. Os parâmetros obtidos a partir dos experimentos são utilizados na realização das análises e das validações dos modelos gerados.

Para obter o tempo de execução de determinada funcionalidade da aplicação móvel, nos experimentos, foram feitas intervenções no código-fonte, utilizando a função *currentTimeMillis()* da classe *System* do Java (ORACLE, 2014b). Esta função retorna o tempo atual do dispositivo em milissegundos. O exemplo de código ilustrado na Figura 4.2 foi utilizado. A partir dos dados obtidos de trechos de códigos como este, foram gerados arquivos de *log* para análise, com os valores do tempo de execução de cada funcionalidade. Vale destacar que é necessário ter acesso ao código-fonte da aplicação para utilizar esse método de obtenção de tempo de execução.

```
1 long tempoInicial = System.currentTimeMills();
2 /*
3  * Alguma operação
4 */
5 long tempoFinal = System.currentTimeMills();
6 long tempoTotal = tempoFinal - tempoInicial; //tempo total de execução da operação
```

Figura 4.2: Exemplo de código em Java utilizado para obter tempo de execução de alguma funcionalidade.

Já para obter os valores relativos ao consumo de energia da aplicação, no presente estudo, foi escolhida a utilização de um software chamado *PowerTutor* (ZHANG et al., 2010). O *PowerTutor* é um componente de gerenciamento de energia que utiliza a geração de modelos para estimar a energia consumida por um processo. Sendo assim, ele informa sobre as implicações do consumo de energia das aplicações do dispositivo móvel. As medições realizadas com o *PowerTutor* foram utilizadas como uma referência para validação das análises realizadas nos modelos analíticos, obtidos a partir da abordagem mostrada neste trabalho.

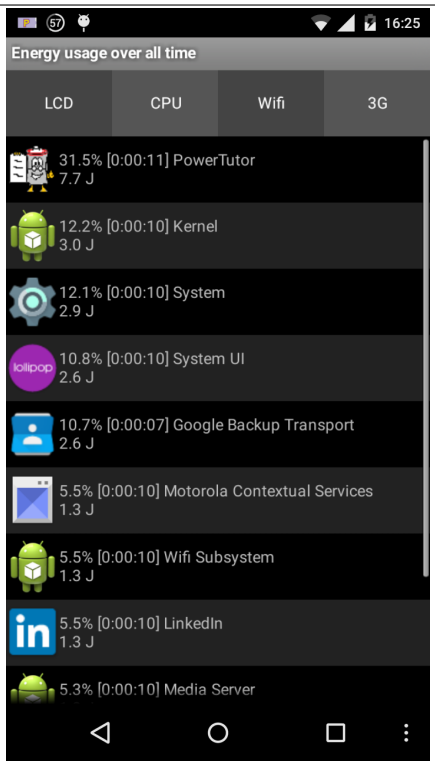


Figura 4.3: *PowerTutor* em execução em dispositivo Android.

O *PowerTutor* apresenta uma característica importante em seu processo de medição do consumo de energia. A medição se dá por processo em execução no dispositivo móvel, ou seja, ele apresenta em seu *log* o consumo de energia de cada processo separadamente. Os registros desse consumo são inseridos no *log* do *PowerTutor* num intervalo de tempo de um segundo, para cada processo. Portanto, para realização das medições de consumo de energia com o *PowerTutor*, cada funcionalidade da aplicação móvel criada para construção do estudo, precisou ser isolada em um único processo no SO do dispositivo móvel. Dessa forma, as funcionalidades da aplicação (captura de imagens, compressão de imagens, *upload*) foram implementadas em processos distintos. Na Figura 4.3 é mostrado o *PowerTutor* em execução.

4.2.1 Experimentos realizados

Experimentos foram conduzidos para se obter os parâmetros de entrada para os modelos gerados utilizando a infraestrutura montada do cenário base. Na configuração do ambiente, foi utilizado um Smartphone Motorola XT1033 (MOBILITY, 2014) com SO Android 4.4.2 com *firmware* original e configurações de desempenho com padrão de fábrica. Nele foi instalada a aplicação móvel a ser avaliada, descrita anteriormente na Seção 4.1. A aplicação foi desenvolvida utilizando a linguagem de programação Java (ORACLE, 2015) para a plataforma *Android*. Durante a execução dos experimentos o *smartphone* estava desconectado do carregador, porém a bateria estava com carga completa no início dos experimentos. A aplicação instalada no *smartphone* consome um serviço que é hospedado em um servidor web. Este servidor web

é alocado em uma VM, presente em um dos oito nós de um *cluster* de uma *cloud* privada Eucalyptus 3.4, que representa a MCC. Para fornecer a conexão WLAN, foi utilizado um *access point* genérico padrão *N* de 150*Megabits por segundo* (Mbps) e o tamanho das imagens utilizadas para realização do *upload* foi de 1 *Megabyte* (MB).

Para os experimentos de *upload* de arquivos, a rede WiFi e WLAN apresentavam sinal de intensidade forte e contavam apenas com um único dispositivo conectado a rede sem fio do *acess point*. Além disso, vale ressaltar que os experimentos de *upload* através da Internet via rede WiFi foram realizados utilizando uma rede doméstica, na qual testes de velocidade apresentaram uma taxa de *upload* inferior a 0,5 Mbps. Já para os experimentos de *upload* através da rede 3G, os testes de velocidade apresentaram uma taxa de *upload* superior a 1Mbps e o sinal apresentava intensidade forte. Por se tratar de um fornecedor público para conectividade 3G, não é possível afirmar a quantidade de dispositivos conectados à rede 3G utilizada no momento dos experimentos.

As medições realizadas devem fornecer um resultado dentro de um intervalo de confiança de 95% (noventa e cinco por cento) com uma margem de erro de 5% (cinco por cento). Para isso, foi utilizada a equação do intervalo de confiança, isolando o número de soluções, para se chegar na quantidade ideal de repetições de experimentos de cada funcionalidade. A equação provê o número de experimentos mínimos necessários para que os valores respeitem o intervalo de confiança e a margem de erro pretendida. Ela é dada pela equação (4.1).

$$n = \left(\frac{Z_{1-\alpha/2} * S}{e\bar{x}} \right)^2 \tag{4.1}$$

Onde, *n* é o número de medições mínimas necessárias para valores dentro de um determinado intervalo de confiança. O valor de $Z_{1-\alpha/2}$ é dado pela tabela da *distribuição-T Student*, de acordo com o intervalo de confiança adotado e o número de medições já realizadas. O desvio padrão é representado pela variável *S* na equação. A variável *e* representa o erro padrão adotado para as medições. E por fim, $\bar{x}$, representa a média das medições iniciais.

Como medições iniciais de todas as funcionalidades (Captura de foto, compressão de imagem e *upload* de imagem) da aplicação, foram executadas vinte repetições, que pela experiência nas medições, mostrou ser uma boa quantidade de medições inicial. Isto foi necessário para se calcular o número mínimo de experimentos realmente necessários, de modo que os resultados ficassem dentro de um intervalo de confiança de 95% e uma margem de erro de 5%. Para a funcionalidade de captura de imagens, por exemplo, nas vinte primeiras repetições realizadas foi obtida uma média de *4105 milissegundos* para a ação, com um desvio padrão de *390,315*. O valor para $Z_{1-\alpha/2}$ na tabela de *distribuição-T Student* para 20 medições, com um intervalo de confiança de 95% e uma margem de erro de 5%, é igual a *2,086*. Portanto, no mínimo serão necessárias *62,94* medições de captura de imagens para que os resultados estejam dentro do limite aceitado. Estes números são sintetizados na Tabela 4.1.

Tabela 4.1: Número mínimo de experimentos necessários para captura de imagens.

Operação	Captura de Fotos
Número de experimentos	20
Tempo Médio de execução	4105 ms
Desvio Padrão	390,315
Exp. necessários	62,94

Dessa forma, têm-se os dados que servem como parâmetros para a execução do modelo, atendendo a uma margem de erro aceitável. Os cálculos do número de experimentos necessários das demais funcionalidades da aplicação móvel são mostrados resumidamente a seguir na Tabela 4.2, para compressão de imagens, e na Tabela 4.3, para *upload* via WLAN.

Tabela 4.2: Número mínimo de experimentos necessários para compressão de imagens.

Tamanho do arquivo	**1MB**
Número de experimentos	20
Média	391,29 ms
Desvio Padrão	146,65 ms
Exp. necessários	884,74

Tabela 4.3: Número mínimo de experimentos necessários para *upload* de imagens via WLAN.

	1MB(.zip)	**1MB(.jpg)**
Num. de exp	20	20
Média	267,2 ms	263,2 ms
Desvio Padrão	62,93	71,11
Exp. necessários	386,23	508,30

Algumas considerações foram adotadas para a realização dos experimentos. (i) Só deveria ser realizado *upload* de uma imagem por vez; (ii) sem falhas de conexão, caso o *access point* falhasse, o experimento deveria ser reiniciado; (iii) sem falhas do *smartphone*, caso o hardware ou o SO do dispositivo móvel falhassem, o experimento seria reiniciado; (iv) sem falhas na MCC, caso algum dos componentes falhassem, o experimento seria reiniciado.

Após a realização da quantidade mínima de experimentos necessários, considerando o intervalo de confiança de 95% e a margem de erro de 5%, foram obtidos os parâmetros de tempo médio de execução e consumo de energia de cada funcionalidade da aplicação móvel. Note que os experimentos foram realizados para arquivos de 1MB. Os resultados são mostrados na Tabela 4.4.

Tabela 4.4: Valores obtidos com os experimentos para arquivos de 1MB

Operação	Tempo médio de execução (*s*)	Consumo médio de energia (*j*)
Captura das Imagens	4.406	94.56
Compressão das imagens	0.335	65.46
Upload via WLAN (.zip)	0.22	7.86
Upload via WLAN (.jpg)	0.266	9.073
Execução completa	**5.0177**	**157.858**

4.3 Modelos gerados

Na abordagem adotada neste trabalho (ver Figura 3.1), modelos são obtidos para representação do cenário de estudo e da aplicação móvel escolhida. A seguir são detalhados os modelos criados para análise do cenário base adotado.

Quanto aos parâmetros utilizados na criação dos modelos, o MTTF para o hardware do dispositivo móvel, não foi encontrado nas especificações do fabricante. Portanto, foram utilizados os valores encontrados em SANDS; TSENG (2010). Já o valor para MTTF do SO do dispositivo móvel (*Android*) foi adaptado dos valores disponíveis em KIM; MACHIDA; TRIVEDI (2009). Para os parâmetros de MTTF e MTTR da aplicação móvel, *access point* e sinal wifi, foram utilizados os valores estimados por OLIVEIRA (2014). Para os componentes da nuvem (máquina física, *hypervisor* KVM, sistemas operacionais e aplicação) foram usados os parâmetros encontrados em DANTAS et al. (2012). Os valores são apresentados na Tabela 4.5. No entanto, por questões de conveniência os parâmetros relacionados entre si foram agrupados. Para isto, foi utilizado análises em RBD utilizando o software Mercury (SILVA, 2013). Os parâmetros agrupados são mostrados na Tabela 4.6.

Tabela 4.5: Parâmetros utilizados nos modelos criados.

Componente	MTTF (h)	MTTR (h)
Hardware disp. móvel (M_HW)	22461,5	1,667
SO móvel (M_SO)	1440,9	0,033
App móvel (M_APP)	336,7	0,0167
Access point (ACP)	10000	1,667
Sinal do WiFi (ACP_S)	6	0,078
Hardware do nó da nuvem (N_HW)	8760	1,667
SO do nó da nuvem (N_SO)	2893	0,25
KVM	2990	1
SO da VM (N_VSO)	2893	0,25
Aplicação da VM (N_VAPP)	788	1

Os agrupamentos dos parâmetros foram realizados utilizando blocos RBD em série, pois nenhum elemento apresenta redundância. Por exemplo, para se obter um único valor de MTTF e MTTR do *access point* (ver Tabela 4.6), levando-se em conta o tempo médio de falha e reparo do *hardware* (ACP) e também o tempo médio de falha e reparo do Sinal do *access point* (ACP_S), foi criado o modelo RBD apresentado na Figura 4.4. Foi realizada uma análise no modelo para

Tabela 4.6: Parâmetros agrupados.

Componente	**MTTF (h)**	**MTTR (h)**
Smartphone (M_HW + M_SO + M_APP)	269,648	0,0395
Access Point (ACP + ACP_S)	5,448	0,0717
MCC (N_HW + N_SO + KVM + N_VSO + N_VAPP)	415,6411	0,8168

obter os valores de MTTF e MTTR relativos ao tempo padrão de um ano (ou 8640 horas) de execução. O resultado da análise mostrou um MTTF de 5,448h e um MTTR de 0,0717h para o conjunto ACP + ACP_S (ver Tabela 4.6).

Figura 4.4: Modelo RBD representando o *access point* e o sinal do *access point*.

Posteriormente então, foi criado foi o primeiro diagrama SysML, o SysML-IBD para representação estática do cenário adotado (ver Figura 4.5). O SysML-IBD é composto por três blocos, cada um representando um elemento do cenário. O primeiro representa o *smartphone*. O segundo o *access point*, que é responsável pela conexão com a MCC. O terceiro bloco representa a MCC.

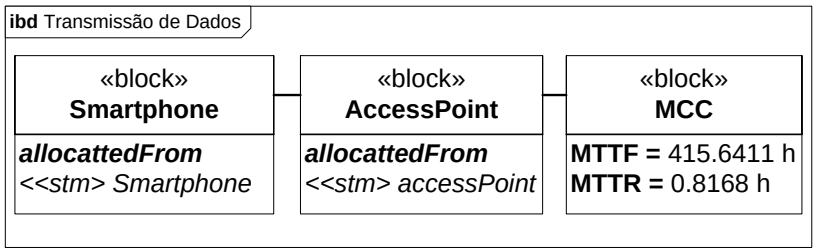


Figura 4.5: SysML-IBD correspondente a Figura 4.1.

Após a criação do SysML-IBD, são criados os SysML-STMs que estão relacionados a cada bloco do SysML-IBD através da marcação de alocação *allocatedFrom*. Dessa forma, foram criados os SysML-STMs do *access point* e do *smartphone*. Visto que o SysML-STM representa o comportamento de um elemento do sistema, ele pode ser construído de várias formas, dependendo das características do elemento que o projetista deseja representar. Nas Figuras 4.6 e 4.7, são apresentados os SysML-STMs criados para o *access point* e o *smartphone*, respectivamente.

O SysML-STM correspondente ao comportamento do *access point* (ver Figura 4.6) é composto por três estados: *Acp_ON*, que indica que o *access point* está funcionando normalmente; *Acp_Falhou*, que indica que houve uma falha, isto é, o access point não está funcionando; e por último o estado *Falha Detectada*, indicando que a falha que causou a indisponibilidade

do *access point* foi detectada e já pode ser reparada. As transições, que são responsáveis pela mudança de estados, estão anotadas com o estereótipo de MARTE *ResourceUsage* e o valor marcado *execTime* para denotar o tempo que essas ações levam para ocorrer. Ademais, o tempo de detecção de falha do *access point* utilizado nesse diagrama, foi estimado.

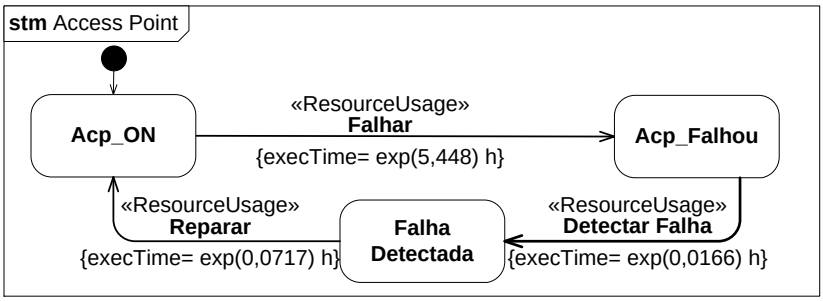
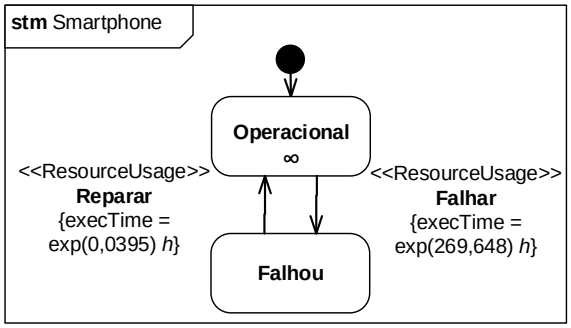


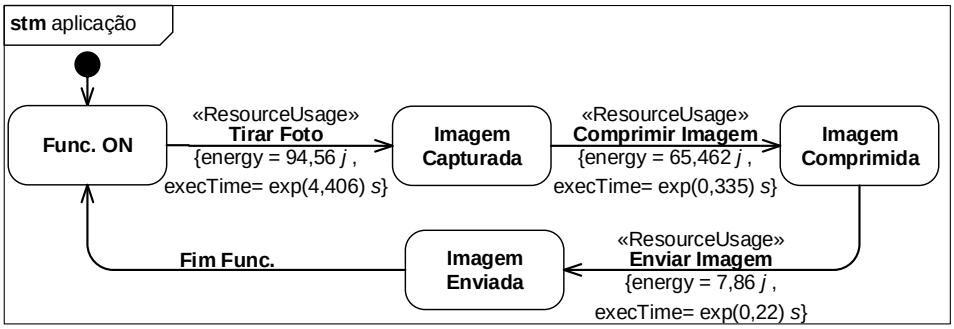
Figura 4.6: SysML-STM correspondente ao *access point*.

Para representar o comportamento do *smartphone* foram criados dois SysML-STMs (ver Figura 4.7), pois o estado *Operacional* (ver Figura 4.7 (a)) é um estado composto. O detalhamento do comportamento do estado composto *Operacional* é mostrado no SysML-STM da Figura 4.7 (b). O *smartphone* tem dois estados possíveis: *Operacional* e *Falho*. Quando o estado *Operacional* estiver ativo, o **smt aplicação** (Figura 4.7 (b)) poderá ser executado. Ele representa o comportamento da aplicação (descrito na Seção 4.1) que é executada no dispositivo móvel. A aplicação inicia no estado *Func. ON*, e assim que o evento *Tirar Foto* é executado, a aplicação alcança o estado *Imagem Capturada*. Após isto, o evento *Comprimir Imagem* pode ser executado e a aplicação chega ao estado *Imagem Comprimida*, representando a ação da compressão de uma imagem pela aplicação. Posteriormente, o evento *Enviar Imagem* pode acontecer e a aplicação alcança o estado *Imagem Enviada*, onde logo em seguida, ocorre o evento *Fim Func.* e a aplicação volta ao seu estado inicial *Func. ON*. Note que as transições do **smt aplicação** (Figura 4.7 (b)) possuem anotações do *profile* MARTE, com o estereótipo *ResourceUsage* e os valores marcados *execTime* e *energy*, denotando o tempo de execução e o consumo médio de energia de cada funcionalidade. Os parâmetros usados de tempo médio de execução e consumo médio de energia, de cada funcionalidade da aplicação, foram atribuídos a partir dos resultados obtidos pela realização de experimentos (ver Seção 4.2.1). Estes resultados são apresentados na Tabela 4.4.

Após a criação dos diagramas da SysML, seguindo a abordagem mostrada anteriormente na Figura 3.1, o próximo passo é obter os modelos DSPNs correspondentes a cada diagrama da SysML seguindo as regras de mapeamento mostradas nas Seções 3.2 e 3.3. Aplicando a regras de mapeamento no SysML-IBD da Figura 4.5, se obtêm o primeiro modelo DSPN (Figura 4.8). Esse modelo DSPN corresponde ao bloco *MCC*, pois este contêm em seus compartimentos de propriedades os valores para o MTTF e MTTR. Por outro lado, os demais blocos (*smartphone* e *access point*) possuem a marcação de alocação *allocatedFrom*, indicando que seu comportamento é descrito por um SysML-STM.



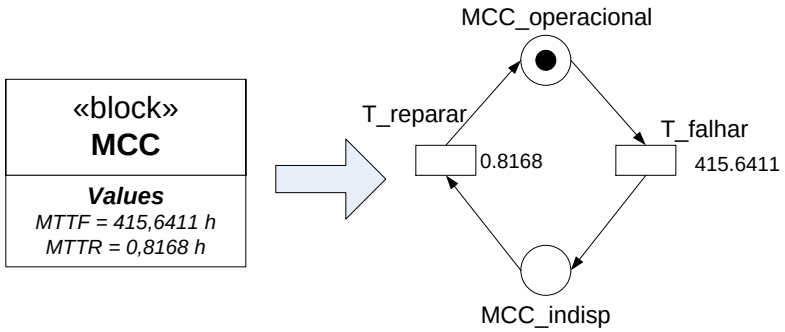
(a) SysML-STM representando o comportamento do *smartphone*



(b) SysML-STM do estado composto, correspondente ao comportamento da aplicação móvel

Figura 4.7: SysML-STMs representando o comportamento do *smartphone* e da aplicação móvel.

No modelo DSPN da Figura 4.8, a presença de um *token* no lugar *MCC_operacional* indica a operacionalidade da MCC. Sendo assim, o serviço que é hospedado na MCC e é consumido pela aplicação móvel também está operacional. Contudo, se alguma falha ocorre, a transição *T_falhar* é disparada, consumindo o *token* do lugar *MCC_operacional*, e gerando um *token* no lugar *MCC_insdisp*. A presença de um *token* no lugar *MCC_insdisp* indica a inoperabilidade da MCC, e consequentemente, a indisponibilidade do serviço nela hospedado. Quando a transição *T_reparar* for disparada, um *token* é gerado no lugar *MCC_operacional*, indicando que a MCC está novamente disponível.



(a) Bloco representando a MCC **(b)** DSPN correspondente ao bloco MCC

Figura 4.8: Mapeamento do bloco MCC.

Já para o **stm *accessPoint*** (Figura 4.6), que representa o comportamento do *access point*, são aplicadas as regras de mapeamento apresentadas na Figura 3.4. Cada estado do SysML-STM é mapeado em lugar na DSPN. As transições que no SysML-STM são anotadas com o valor marcado *execTime* de MARTE, foram mapeadas em transições exponenciais na DSPN. Os respectivos valores atribuídos foram adotados como *delays* para as transições exponenciais. A DSPN resultante do processo de mapeamento é mostrada na Figura 4.9.

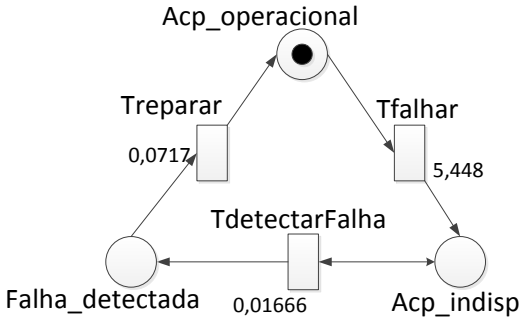
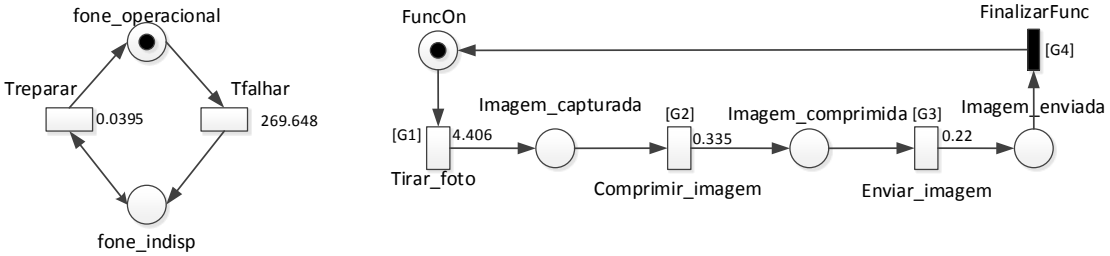


Figura 4.9: DSPN correspondente ao **stm *access point*** mostrado na Figura 4.6.

No modelo DSPN da Figura 4.9, quando um *token* está presente no lugar *Acp_operacional*, indica que o *access point* está funcionando normalmente, e é capaz de transmitir dados entre o dispositivo móvel e a MCC. Quando a transição *Tfalhar* é disparada, o *token* é consumido pela transição e depositado no lugar *Acp_indisp*, indicando que o *access point* se encontra indisponível. Posteriormente, a transição *TdetectarFalha* pode ser disparada e o *token* alcança o lugar *Falha_detectada*. A partir daí, a transição *Treparar* pode ser disparada, para que o *access point* volte a funcionar normalmente. Quando a essa transição é disparada, o *token* é consumido pela transição e volta ao lugar *Acp_operacional*, indicando que o *access point* já está operando normalmente.

No mapeamento dos SysML-STMs que representam o comportamento do *smartphone* foram obtidas duas DSPNs, cada uma correspondente a um SysML-STM que é mostrado na Figura 4.7. A DSPN correspondente ao diagrama **stm *Smartphone*** (Figura 4.7 (a)) é mostrada na Figura 4.10 (a). Os estados foram mapeados em lugares na DSPN. As transições anotadas com o *profile* MARTE foram mapeadas em transições exponenciais e atribuídos os respectivos *delays*. Semelhantemente, o mesmo ocorre para DSPN que representa o comportamento do estado composto (Figura 4.10 (b)). Contudo, as transições desta última são anotadas com os valores marcados *energy* e *execTime* do *profile* MARTE, indicando a energia média e o tempo médio de execução que é considerado por cada evento da aplicação móvel. Dessa forma, através de análises no modelo DSPN é possível obter o consumo de energia estimado da aplicação. As equações utilizadas para estimar o consumo médio de energia e o tempo médio de execução são mostradas na Tabela 4.7. As equações foram obtidas seguindo a mesma abordagem explicada anteriormente na Seção 3.3. Na equação para obtenção do tempo médio de execução é calculado

o tempo médio de absorção do modelo, representando o inverso do *throughput* da transição *Tirar_Foto*. A equação da métrica de estimativa consumo de energia total é composta pela soma da estimativa de consumo de energia de cada função/atividade da aplicação. Note também que as funções de guarda (G1, G2, G3 e G4) que são atribuídas pelo projetista da aplicação com a função de verificar se o estado *Operacional* está ativo, são utilizadas em todas as transições. A função de guardas utilizada no modelo é detalhadas na Tabela 4.8.



(a) DSPN correspondente ao **smt Smartphone** **(b)** DSPN correspondente ao **stm aplicação**

Figura 4.10: DSPNs correspondentes aos SysML-STMs da Figura 4.7.

Tabela 4.7: Funções do TimeNET utilizadas para obtenção das métricas.

Métrica	**Função**
Tempo médio de execução	$1/((P\{\#FuncOn > 0\}) * (1/4.406))$
Consumo de energia	$P\{\#FuncOn > 0\} * 1/4.406) * 94.56 * 4.972) + ((P\{\#Imagem\_capturada > 0\} * 1/0.335) * 65.462 * 4.972) + ((P\{\#Imagem\_comprimida > 0\} * 1/0.22) * 7.86 * 4.972)$

Na execução da DSPN mostrada na Figura 4.10 (a), quando um *token* está presente no lugar *fone_operacional* indica que o *smartphone* está funcionando. Porém, se a transição exponencial *Tfalhar* for disparada, um *token* é retirado do lugar *fone_operacional* e um *token* é depositado no lugar *fone_indisp*, indicando que o smartphone falhou e se encontra indisponível. Quando a transição *Treparar* for disparada, o *token* volta o lugar *fone_operacional*, indicando que o *smartphone* já está funcionando novamente.

Já no modelo DSPN da Figura 4.10 (b), a presença de um *token* no lugar *FuncOn* indica que a aplicação foi iniciada e está pronta para uso. A transição *Tirar_foto* só pode ser disparada quando a função de guarda *G1* for satisfeita, o que ocorre quando um *token* está presente no lugar *fone_operacional* da DSPN da Figura 4.10 (a). Quando a transição *Tirar_foto* é disparada, um *token* é gerado no lugar *Imagem_capturada*, denotando que a foto tirada pelo usuário através da aplicação, e já foi armazenada no dispositivo móvel. Posteriormente, quando a função de guarda *G2* é satisfeita, a transição *Comprimir_imagem* é disparada. Então um *token* é depositado no lugar *Imagem_comprimida*, indicando que a imagem que capturada foi comprimida e está pronta para ser enviada. Quando a função de guarda *G3* é satisfeita, a transição *Enviar_imagem* é disparada. Sendo assim, um *token* é gerado no lugar *Imagem_enviada* indicando que o *upload* da

imagem foi realizado. Em seguida, se a função de guarda *G4* for satisfeita, a transição imediata *FinalizarFunc* será disparada, e um *token* é depositado no lugar *FuncOn*, indicando que já está pronta para mais uma execução.

Tabela 4.8: Função de guarda utilizada na DSPN da Figura 4.10 (b).

Guarda	**Função**
G1, G2, G3, G4	#fone_operacional = 1

Dessa forma, todos os modelos para representar a execução da aplicação do cenário base foi concluída. A próxima fase do processo é realizar as análises nas DSPNs para obter as métricas relativas a desempenho, consumo de energia e disponibilidade.

4.4 Considerações Finais

Neste capítulo foi apresentado o cenário base escolhido para estudo, bem como o método de obtenção dos parâmetros de tempo médio de execução e consumo médio de energia, os quais foram utilizados nos modelos gerados. O parâmetro de tempo de execução foi obtido através de intervenções no código-fonte da aplicação móvel, enquanto o parâmetro de consumo médio de energia foi obtido através da realização de experimentos utilizando o software *PowerTutor*. Também foi apresentada a metodologia de condução dos experimentos para obtenção destes parâmetros. Posteriormente a realização dos experimentos, foram obtidos os diagramas SysML para realização do estudo e geradas as DSPNs através do processo de mapeamento dos diagramas da SysML.

5

Estudo de Casos

The significant problems we have, cannot be solved at the same level of thinking with which we created them.

—ALBERT EINSTEIN

Neste capítulo são apresentados os estudos de casos realizados para avaliação da abordagem empregada, além da validação das DSPNs resultantes do mapeamento dos diagramas SysML. Primeiro é apresentado como a validação dos modelos foi realizada. No primeiro estudo caso, é adicionado o 3G como conexão de rede, para que a aplicação móvel possa realizar o *upload* de arquivos também via rede 3G. No segundo estudo de caso, é utilizada uma arquitetura de *cloudlet* para investigar os efeitos desse tipo de arquitetura em um cenário que contenha uma aplicação móvel. Por fim, é realizado um comparativo entre os estudos de caso e o cenário base. Também é realizada uma variação de cenário, onde é proposta uma mudança funcional na aplicação móvel, a fim de observar os efeitos causados nas métricas de desempenho (tempo de execução), consumo de energia e disponibilidade.

5.1 Validação dos Modelos

A validação dos modelos DSPNs gerados pelo processo de mapeamento, tem como objetivo mostrar que os resultados obtidos através das análises estocásticas são estatisticamente próximos aos resultados obtidos por um experimento em uma infraestrutura real. Para execução dos modelos DSPNs, foram utilizados os parâmetros obtidos através da realização dos experimentos. Estes parâmetros são exibidos na Tabela 4.4. As análises foram realizadas com o auxílio da ferramenta TimeNET (ZIMMERMANN; KNOKE, 2007).

Durante a execução dos experimentos foram consideradas algumas restrições. Essas restrições envolviam controle de falha para conexões, *hardware* e *software* dos elementos utilizados. Durante a análise dos modelos DSPN isto também teve de ser considerado. Dessa forma, apenas a DSPN da Figura 4.10 (b), foi utilizada para fins de comparação. Realizando uma

simulação na DSPN, com resultados obedecendo a um intervalo de confiança de 95% e margem de erro de 5%, foi obtido um tempo de execução médio de 4.972 segundos, mas os valores dentro do intervalo de confiança variam entre 4.8026 segundos e 5.1546 segundos. Uma análise estacionária também foi realizada no modelo DSPN, apresentando um tempo de execução de 4.961 segundos. No experimento, obedecendo ao mesmo intervalo de confiança e margem de erro, foi obtido um tempo de execução médio entre 4.7945 segundos e 5.2408 segundos (ver Figura 5.1). Para obter os valores do experimento que estão dentro do intervalo de confiança desejado e margem de erro pretendida, foi usado o software MiniTab (MINITAB, 2015). Nele foram inseridos todos os *logs* dos experimentos realizados no cenário base e calculado os valores pertencentes ao intervalo de confiança, através da função de resumo estatístico. Dessa forma, fazendo comparações estatísticas, é possível mostrar que os resultados obtidos a partir da simulação da DSPN apresentam uma boa aproximação em relação às medições realizadas na infraestrutura real. Entretanto, o TimeNET não disponibiliza os valores utilizados no processo de simulação. Dessa forma, não foi possível realizar testes estatísticos para provar que os valores utilizados nas transições durante a simulação da DSPN não possuem diferença estatística se comparados aos valores dos experimentos.

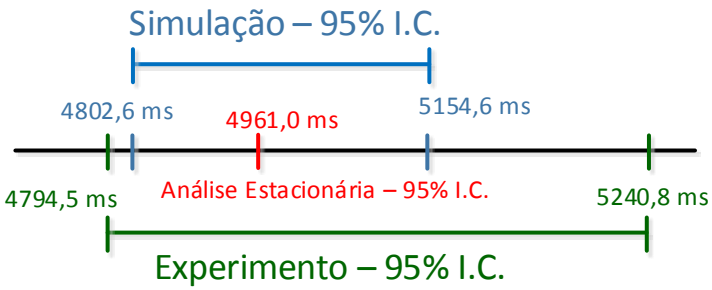


Figura 5.1: Resultados de tempo médio de execução dos experimentos e da simulação da DSPN

Com relação ao consumo de energia, os resultados também apresentaram uma boa aproximação. Contudo, não foi possível calcular o intervalo de confiança para as medições de consumo de energia da aplicação móvel. Os experimentos foram realizados em "blocos"de arquivos, o que impossibilitou as medições caso a caso. Por exemplo, eram realizados 100 *uploads* de imagem, uma imagem por vez, mas a energia consumida só era computada ao final do envio das 100 imagens. Posteriormente era calculada a média do consumo de energia por arquivo. Dessa forma, o número de amostras não foi suficiente para demonstrar a variação da média num intervalo de confiança. Sendo assim, nos experimentos, o valor médio para o consumo de energia foi de 157.858 *joules*. Já nos resultados da simulação da DSPN, respeitando o mesmo intervalo de confiança (95%) a e margem de erro (5%), o valor se apresentou entre 152.723 *joules* e 165.620 *joules*. Esses valores são muito próximos a média obtida na infraestrutura real, sendo essa diferença de 5% para mais ou para menos. As funções utilizadas no TimeNET para obter as

estimativas das métricas de tempo médio de execução, e de consumo de energia são exibidas na Tabela 5.1. A função para obter a métrica de tempo médio de execução é calculada através do tempo de absorção do modelo, onde é utilizada o valor da transição *capturar_foto* (4.406). Para obter a métrica de consumo de energia é realizado a soma da estimativa de consumo de energia de cada funcionalidade da aplicação. O tempo de observação (t) utilizado, é dado pela média obtida nos resultados da métrica de tempo de execução na simulação do modelo DSPN (4.972 segundos).

Tabela 5.1: Funções do TimeNET utilizadas para obtenção das métricas.

Métrica	**Função**
Tempo médio de execução	$1/((P\{\#FuncOn > 0\}) * (1/4.406))$
Consumo de energia	$((P\{\#FuncOn > 0\} * 1/4.406) * 94.56 * 4.972) + ((P\{\#Imagem\_capturada > 0\} * 1/0.335) * 65.462 * 4.972) + ((P\{\#Imagem\_comprimida > 0\} * 1/0.22) * 7.86 * 4.972)$

5.2 Estudo de Caso 1: *Upload* de arquivos com múltiplas interfaces de rede

É comum nos dias atuais que os dispositivos móveis (*smartphones*, *tablets*, etc.) apresentem várias interfaces de rede (WiFi, 3G, *bluetooth*, etc.) para comunicação com outros dispositivos. Isso aumenta o tempo de conectividade dos usuários ajudando-os a ficar *on-line* por um tempo maior. O crescimento do tráfego pelas redes móveis já é alto e estima-se que esse tráfego cresça ainda mais nos próximos anos (INDEX, 2015). Contudo, as conexões de redes móveis (ex., 3G) não apresentam uma boa eficiência energética, influenciando na descarga mais rápida da bateria do dispositivo. Isso torna a preocupação com o consumo de energia do aparelho, por meio de conexões de redes móveis, ainda mais efetiva.

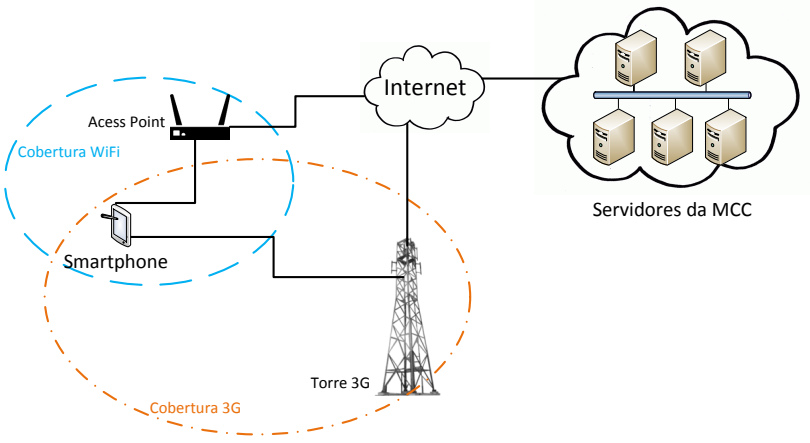


Figura 5.2: Cenário ilustrando *upload* de imagens via diferentes interfaces de rede.

Neste estudo de caso, consideramos uma aplicação capaz de realizar *upload* de imagens para servidores alocados em uma MCC através de diferentes interfaces de rede, para investigar se a utilização de mais de uma interface de rede pode apresentar ganhos, principalmente de desempenho (tempo de execução) para a aplicação móvel. No cenário, a aplicação que está instalada em um dispositivo móvel, utiliza a Internet para consumir um serviço que é hospedado em um servidor. O servidor é alocado em uma nuvem privada Eucalyptus, representando uma MCC. No comportamento da aplicação, a realização do *upload* das imagens pode ocorrer através da interface WiFi ou 3G. Porém, a rede WiFi tem preferência para comunicação, o que significa que, o *upload* via rede móvel 3G só irá ocorrer caso a rede WiFi esteja indisponível. A Figura 5.2 ilustra um cenário no qual o dispositivo móvel pode se comunicar com uma MCC através de interface de rede WiFi ou da interface 3G.

5.2.1 Criação dos diagramas SysML e Mapeamento em DSPNs

Para realização das análises de desempenho e consumo de energia do estudo de caso, foram criados os diagramas da SysML correspondentes à infraestrutura e aplicação utilizada. Na Figura 5.3, é apresentado o SysML-IBD, que representa a configuração estática do sistema e a relação entre seus elementos. Cada bloco no SysML-IBD representa um elemento do sistema. Os blocos do *smartphone* e do *access point* são anotados com a propriedade *allocatedFrom*, indicando que o comportamento de cada um desses blocos será detalhado por um SysML-STM. Os demais blocos, MCC e Torre de Sinal 3G, apresentam em seu compartimento de propriedades os valores para os MTTFs e MTTRs, indicando o comportamento de falha e reparo desses elementos.

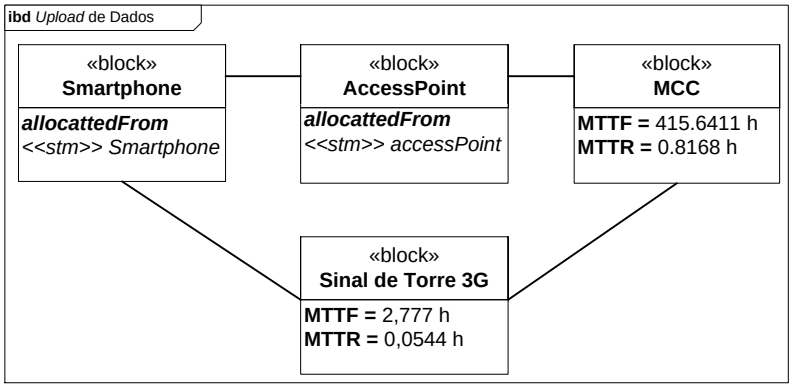


Figura 5.3: SysML-IBD correspondente a Figura 5.2.

As Figuras 5.4 e 5.5 apresentam os comportamentos do *access point* e do *smartphone*, respectivamente, em termos de estados pelos SysML-STMs. O *access point* tem como função conectar o *smartphone* a Internet para transmitir os dados da aplicação móvel aos servidores. Seu SysML-STM apresenta três estados possíveis. *Acp_ON* representa o *access point* funcionando normalmente, fornecendo sinal para conectividade com a Internet. *Acp_Falhou* é o estado em que

o *access point* está indisponível, não é possível fornecer conexão à Internet para o *smartphone*. O estado *Falha Detectada* representa o estado em que a falha que causou a indisponibilidade do *access point* foi detectada.

Já para o *smartphone*, existem dois SysML-STMs (ver Figura 5.5). No SysML-STM apresentado na Figura 5.5 (b), representa o comportamento da aplicação que é executada no dispositivo móvel, sendo o detalhamento do estado composto *Operacional*. Note que nesse SysML-STM é utilizada a estrutura de escolha para verificar se a conexão WiFi está disponível, pois caso ela não esteja, o *upload* da imagem será realizado através da rede móvel 3G. Contudo, caso 3G não também não esteja disponível, a aplicação aguarda a reativação da rede 3G para envio do arquivo.

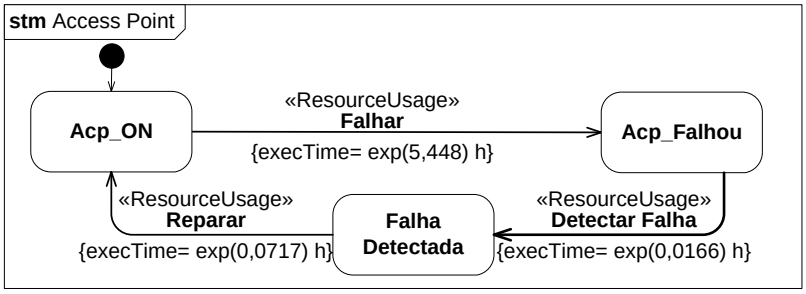
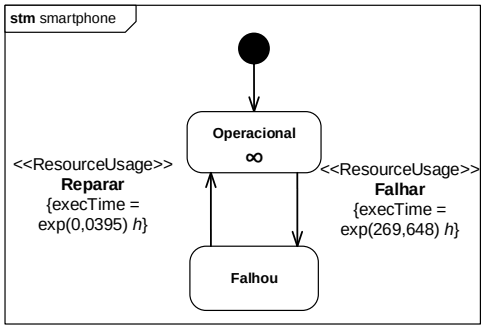
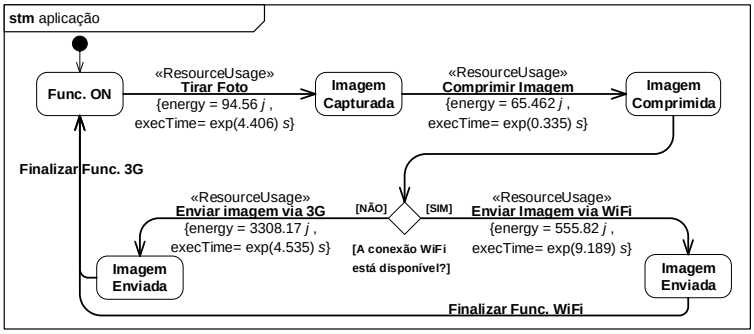


Figura 5.4: SysML-STM correspondente ao comportamento do *access point*.



(a) SysML-STM representando o comportamento do smartphone



(b) SysML-STM representando o comportamento da aplicação móvel

Figura 5.5: SysML-STMs representando o comportamento do *smartphone*, e o detalhamento do estado composto representando o comportamento da aplicação móvel

Uma vez criados os diagramas SysML que representam o cenário de estudo, é realizado o mapeamento. As DSPNs resultantes do processo de mapeamento são exibidas na Figura 5.6.

Primeiramente foi realizado o mapeamento do SysML-IBD (ver Figura 5.3). Para esse modelo, inicialmente foram obtidas as DSPNs correspondentes ao bloco da *MCC* (ver Figura 5.6 (c)) e da *Torre de Sinal 3G* (ver Figura 5.6 (e)), pois esses dois blocos do SysML-IBD apresentam em seus compartimentos de propriedades os valores de seus respectivos MTTF e MTTR. Posteriormente, os SysML-STMs associados a cada bloco, foram mapeados em DSPNs. A DSPN correspondente ao SysML-STM que representa o *access point* é mostrada na Figura 5.6 (d), enquanto a DSPN referente ao **stm *smartphone*** é exibida na Figura 5.6 (b). Note que no **stm *smartphone*** o estado *Operacional* é um estado composto. O mapeamento desse estado composto é exibido na Figura 5.6 (a). Nesse último, é possível notar a presença das funções de guarda, as quais são detalhadas na Tabela 5.2.

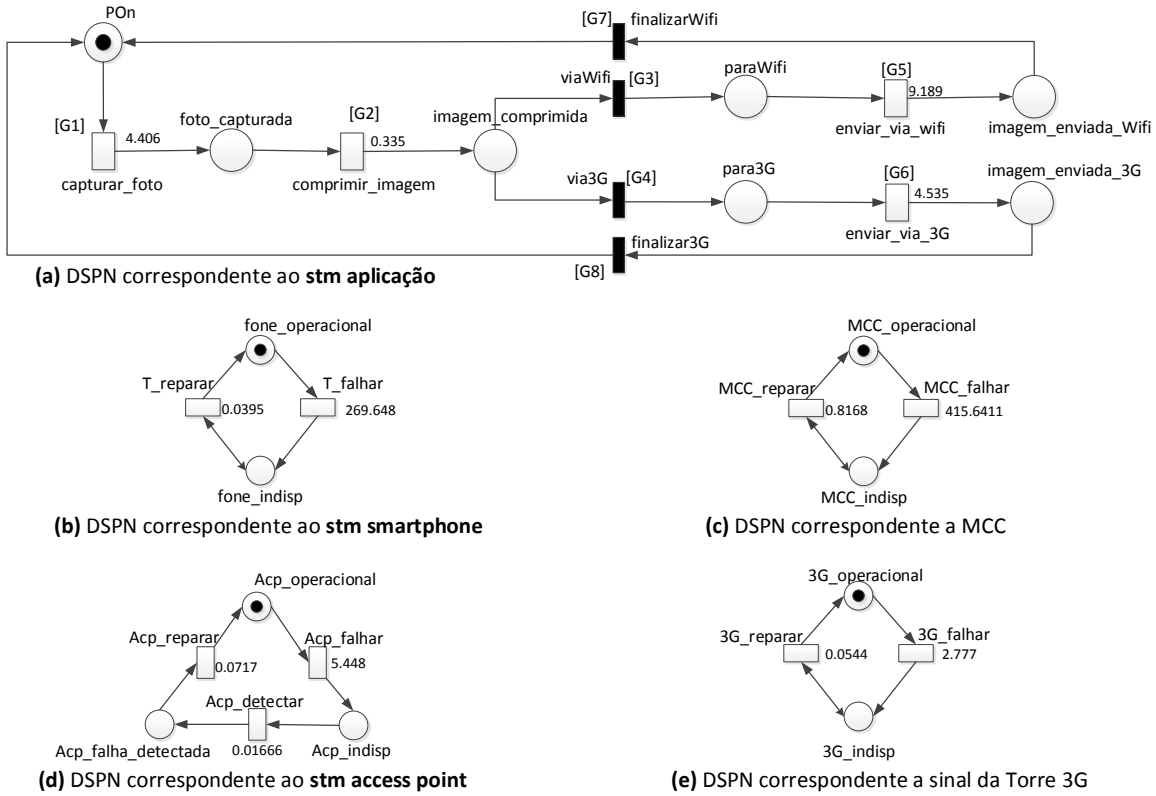


Figura 5.6: DSPNs obtidas a partir do mapeamento dos diagramas SysML criados.

As funções de guarda que contém a função $\#fone\_operacional = 1$, foram obtidas através do mapeamento do SysML-STM com estados compostos. Essa função faz a verificação se o modelo DSPN da Figura 5.6 (b) está operacional. A função de G3, além de fazer essa verificação, também analisa se o *acess point* (Figura 5.6 (d)) e a MCC (Figura 5.6 (c)) estão operacionais, para que o *upload* possa ser realizado via rede WiFi. Da mesma forma, a função G4 faz outras verificações além da operacionalidade do modelo DSPN da Figura 5.6 (b). Ela verifica se a Torre de Sinal 3G (Figura 5.6 (e)) está operacional para realização do *upload* via

rede 3G, se a MCC também está operacional e se o *access point* está inoperante.

Tabela 5.2: Funções de guarda da DSPN mostrada na Figura 5.6 (a).

Guarda	Função
G1	#fone_operacional = 1
G2	#fone_operacional = 1
G3	#fone_operacional = 1 AND #Acp_operacional = 1 AND #MCC_operacional = 1
G4	#fone_operacional = 1 AND #Acp_operacional = 0 AND #MCC_operacional = 1 AND #3G_operacional = 1
G5	#fone_operacional = 1
G6	#fone_operacional = 1
G7	#fone_operacional = 1
G8	#fone_operacional = 1

5.2.2 Análises numéricas

Nesta seção, são mostrados os resultados obtidos a partir de análises realizadas nas DSPNs geradas a partir do processo de mapeamento. Todos os valores utilizados que estão relacionados a tempo, são considerados exponencialmente distribuídos, para uma estimativa mais real das métricas. Os parâmetros de MTTF e MTTR utilizados nas DSPNs são listados na Tabela 4.6.

Além destes parâmetros, os valores para o MTTF e MTTR do Sinal da Torre 3G foram obtidos em COOPER; FARRELL (2007). Outros parâmetros que são utilizados na DSPN da Figura 5.6 (a), para tempo médio de execução das operações realizadas pela aplicação móvel, bem como para o consumo médio de energia associado a estas operações, foram obtidos através de experimentos prévios (ver Seção 4.2.1). Os valores listados correspondem a operações que utilizam arquivos de 1MB. A Tabela 5.3 apresenta estes parâmetros.

Tabela 5.3: Parâmetros de tempo médio de execução e consumo médio de energia.

Operação	Tempo médio de execução (s)	Consumo médio de energia (*j*)
Captura de Foto	4,406	94,56
Compressão de Imagens	0,335	65,462
Upload via WiFi	9,189	555,82
Upload via 3G	4,535	3308,17

Foram realizadas análises estacionárias nos modelos gerados, a partir do mapeamento (ver Figura 5.6) através do TimeNET. Vale ressaltar que nas análises não foram considerados o tempo e consumo de energia para o chaveamento para ativação/desativação das interfaces WiFi e 3G do dispositivo móvel. Os resultados mostram uma média de consumo de energia de 589,122 *joules* para a aplicação móvel e um tempo médio de execução de 11,144 segundos. Nestas condições, a análise também mostrou que o sistema completo apresenta uma disponibilidade de

99,777%. A alta disponibilidade se dá principalmente pela presença da rede 3G, que pode estar disponível para uso, caso a rede WiFi fique indisponível. Contudo, o alto consumo de energia da aplicação, pode ter relação também com a utilização da rede 3G. As funções utilizadas no TimeNET para obter as estimativas das métricas de tempo médio de execução, consumo de energia e disponibilidade são exibidas na Tabela 5.4. O tempo médio de execução é obtido através da equação do tempo de absorção do modelo, onde a transição relacionada é a *capturar_foto*, por isso o tempo associado a transição está presente na equação (4.406). Já a métrica de consumo de energia utiliza os valores de consumo de energia, obtidos através de experimentos que estão associados a cada funcionalidade da aplicação (explicitados no SysML-STM da Figura 5.5). É realizada a soma dos valores de estimativas de cada funcionalidade para se obter a estimativa total do consumo de energia. Por fim, a equação que calcula a métrica da disponibilidade verifica a probabilidade de todo o sistema está operacional, em ao menos um dos fluxos de *upload*, ou por WiFi ou via rede 3G.

Tabela 5.4: Equações do TimeNET utilizadas para obtenção das métricas do estudo de caso.

Métrica	**Função**
Tempo médio de execução	$1/((P\{\#POn > 0\}) * (1/4.406))$
Consumo de energia	$((P\{\#POn > 0\} * 1/4.406) * 94.56 * 11.144) + ((P\{\#foto\_capturada > 0\} * 1/0.335) * 65.462 * 11.144) + ((P\{\#paraWiFi > 0\} * 1/9.189) * 555.82 * 11.144) + ((P\{\#para3G > 0\} * 1/4.535) * 3308.17 * 11.144)$
Disponibilidade	$P\{\#POn > 0 \ AND \ \#MCC\_operacional > 0 \ AND \ (\#3G\_operacional > 0 \ OR \ \#Acp\_operracional > 0)\}$

5.3 Estudo de Caso 2: *Upload* de arquivos com múltiplas interfaces de rede e *Cloudlet*

No segundo estudo de caso, foi adotado o conceito de *cloudlet* para investigar se o uso deste tipo de arquitetura poderia apresentar ganhos, principalmente no que se refere ao consumo de energia para a aplicação desenvolvida. Uma *cloudlet* é uma infraestrutura computacional que está mais próxima aos usuários (ex., shoppings, aeroportos), conectada a Internet e que dispõe de bons recursos computacionais disponíveis para uso (SATYANARAYANAN et al., 2009). A utilização desse tipo de arquitetura, muitas vezes, está relacionada com a tentativa de aumento da disponibilidade de serviços que são consumidos pelas aplicações, visto que as aplicações se tornam mais acessíveis aos usuários que as utilizam. Além disso, podem fornecer um tempo de resposta mais baixo para execução dos serviços, por estar mais próxima ao consumidor destes serviços. Dessa forma, foi criado o cenário mostrado na Figura 5.7 para avaliação.

No cenário mostrado na Figura 5.7, o cliente que utiliza a aplicação no dispositivo móvel pode efetuar o *upload* de arquivos para os servidores da *cloudlet*, caso a conexão WLAN esteja

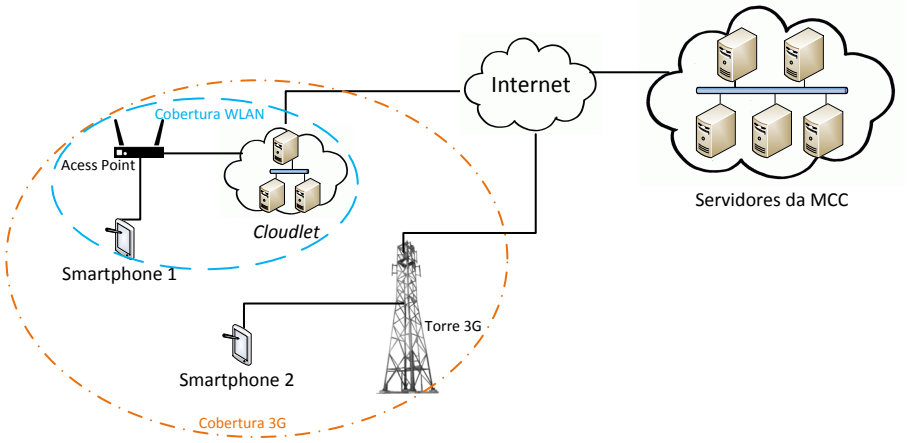


Figura 5.7: Cenário adotado para estudo, contendo uma arquitetura de *cloudlet*.

disponível para ele. Isto simula um local de trabalho, por exemplo, onde há conexão WLAN para funcionários e clientes. Nesse caso, a *cloudlet* é encarregada de fazer a sincronização dos arquivos com os servidores da MCC. Entretanto, se o dispositivo móvel não estiver dentro da área de cobertura da WLAN, a aplicação poderá realizar o *upload* do arquivo via rede 3G diretamente para os servidores localizados na MCC.

5.3.1 Criação dos diagramas SysML e Mapeamento em DSPNs

Na primeira etapa do processo, foram criados os diagramas da SysML para representar o cenário escolhido para análise. No SysML-IBD mostrado na Figura 5.8, é apresentado a visão estática do sistema. Note que o bloco que representa a Torre de sinal 3G possui conexão apenas com o bloco do *Smartphone* e o bloco da MCC. Isto indica que, para este exemplo, via rede 3G o *Smartphone* não pode acessar a *cloudlet* diretamente.

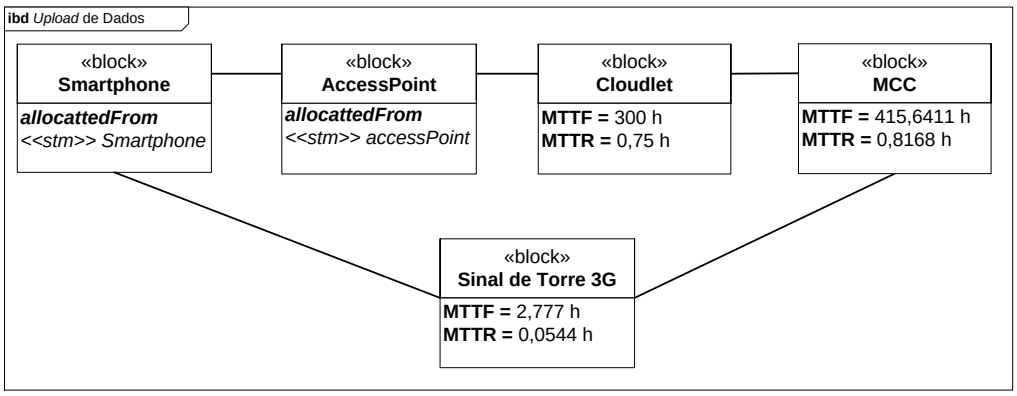


Figura 5.8: SysML-IBD representando o cenário da Figura 5.7.

Posteriormente a criação do SysML-IBD que representa o cenário adotado neste estudo de caso, foram criados os SysML-STMs referentes ao *access point* e ao *smartphone*. Estes dois diagramas são similares aos mostrados nas Figuras 5.4 e 5.5, com variação de alguns

parâmetros, relacionados ao tempo médio de *upload* via rede WLAN e ao consumo de energia dessa operação.

O mapeamento dos diagramas da SysML do cenário da Figura 5.7, resultou em seis DSPNs. Todas elas são exibidas na Figura 5.9. No mapeamento do SysML-IBD, foram obtidas as DSPNs correspondentes aos blocos da *Cloudlet* (ver Figura 5.9 (d)), da MCC (ver Figura 5.9 (c)) e da Torre de Sinal 3G (ver Figura 5.9 (f)), pois esses blocos possuem em seus compartimentos de propriedades os valores de seus respectivos MTTF e MTTR. O SysML-STM correspondente ao *access point* foi mapeado na DSPN mostrada na Figura 5.9 (e), enquanto o SysML-STM que representa o *smartphone* foi mapeado na DSPN da Figura 5.9 (b). Já seu estado composto *Operacional* foi mapeado na DSPN da Figura 5.9 (a). Vale ressaltar que a diferença entre o modelo DSPN da Figura 5.6 (a) e o modelo DSPN da Figura 5.9 pode ser notado na transição *enviar_via_wlan* da Figura 5.9, que apresenta um tempo associado e uma nomenclatura diferente.

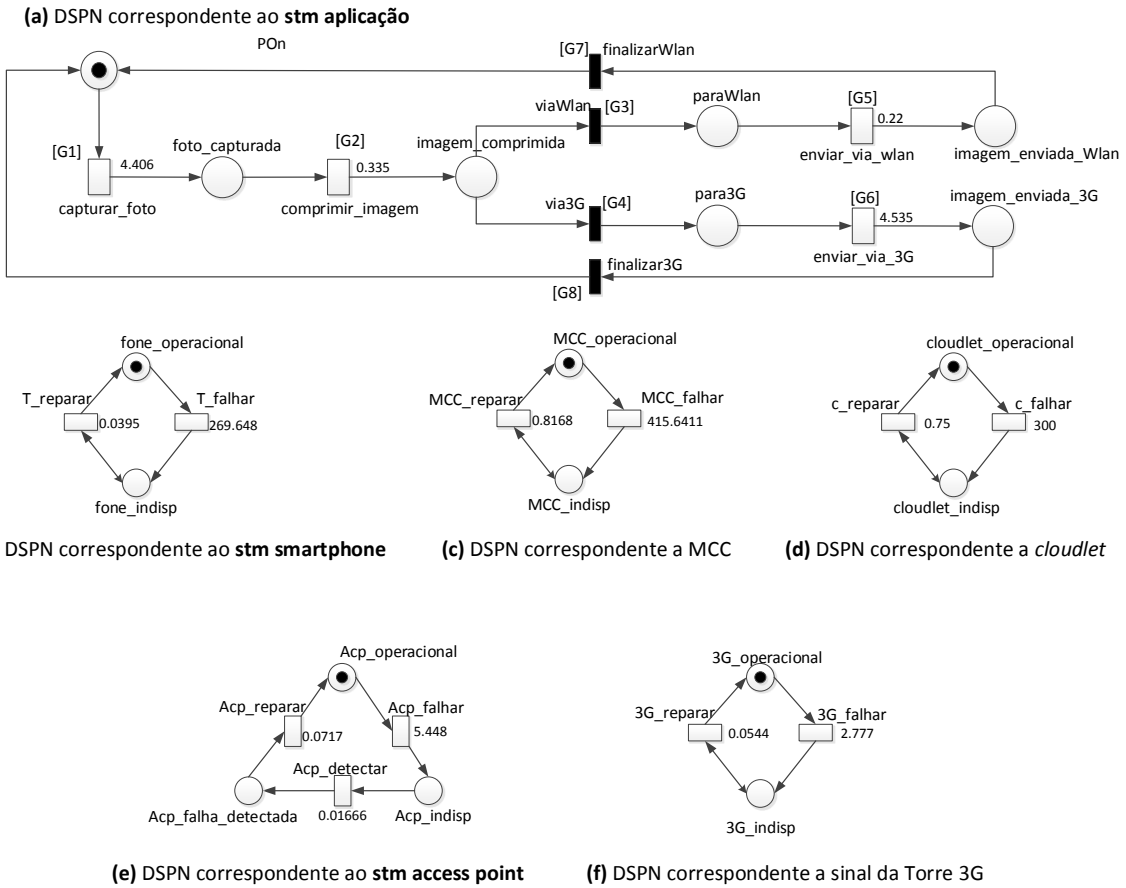


Figura 5.9: DSPNs obtidas a partir do mapeamento dos diagramas SysML criados.

Com a utilização da arquitetura com *cloudlet* para realização do *upload* de imagens via rede WLAN, a DSPN deve verificar se o *access point* está disponível para comunicação e se a *cloudlet* está disponível para aceitar essa conexão. Devido a tal condição, funções de guarda foram utilizadas. Na Figura 5.9 (a) é possível notar a presença de oito delas. Todas têm suas funções descritas na Tabela 5.5. As funções de guarda que contém a função $\#fone\_operacional = 1$,

foram obtidas através do mapeamento do SysML-STM com estados compostos. Essa função faz a verificação se o modelo DSPN da Figura 5.9 (b) está operacional. A função de G3, além de fazer essa verificação, também analisa se o *acess point* (Figura 5.9 (e)) e a *cloudlet* (Figura 5.9 (d)) estão operacionais, para que o *upload* possa ser realizado via rede WLAN. Já a função G4, além de verificar a operacionalidade do modelo DSPN da Figura 5.9 (b), verifica se a Torre de Sinal 3G (Figura 5.9 (f)) e se a MCC (Figura 5.6 (c)) estão operacionais. Isso para garantir a realização do *upload* via rede 3G, caso o *access point* esteja ou a *cloudlet* estejam inoperantes.

Tabela 5.5: Funções de guarda da DSPN mostrada na Figura 5.9 (a).

Guarda	**Função**
G1	#fone_operacional = 1
G2	#fone_operacional = 1
G3	#fone_operacional = 1 AND #Acp_operacional = 1 AND #cloudlet_operacional = 1
G4	#fone_operacional = 1 AND #Acp_operacional = 0 AND #MCC_operacional = 1 AND #3G_operacional = 1
G5	#fone_operacional = 1
G6	#fone_operacional = 1
G7	#fone_operacional = 1
G8	#fone_operacional = 1

5.3.2 Análises numéricas

Após a obtenção das DSPNs através do processo de mapeamento, então o cenário foi analisado. Os parâmetros utilizados para MTTF e MTTR dos elementos do dispositivo móvel, MCC, *access point* são apresentados na Tabela 4.6. Os valores do MTTF e MTTR para o *Sinal da Torre 3G* foram obtidos em COOPER; FARRELL (2007). Já os valores do MTTF e MTTR da *cloudlet* foram estimados baseados em uma arquitetura de nuvem privada que possua menos recursos do que uma nuvem, que neste estudo de caso é representado pelo bloco da MCC. É importante notar que esses valores que são atribuídos as DSPNs são previamente definidos nos SysML-IBD e SysML-STMs, através das anotações do *profile* MARTE. A Tabela 5.6 mostra os parâmetros das operações da aplicação móvel que tem seu modelo DSPN apresentado na Figura 5.9 (a).

Tabela 5.6: Parâmetros de tempo médio de execução e consumo médio de energia do estudo de caso.

Operação	**Tempo médio de execução (s)**	**Consumo médio de energia (*j*)**
Captura de Foto	4,406	94,56
Compressão de Imagens	0,335	65,462
Upload via WLAN	0,22	7,86
Upload via 3G	4,535	3308,17

As análises realizadas nos modelos DSPN da Figura 5.9 estimaram um tempo médio

de execução de 4,953 segundos para aplicação móvel, com um consumo médio de energia de 178,755 joules. Além disso, a disponibilidade do sistema foi estimada em 99,874%, isto é, apresentou a maior disponibilidade dos cenários analisados, até o momento. Dessa forma, os resultados estimados mostram que a utilização de uma arquitetura de *cloudlet* no sistema pode de fato ajudar na disponibilidade do sistema e representar um ganho no consumo de energia, se comparado ao cenário apresentado no estudo de caso 1. As funções utilizadas no TimeNET para obter as estimativas das métricas de tempo médio de execução, consumo de energia e disponibilidade são exibidas na Tabela 5.7. Para obtenção de tempo médio de execução é usada a equação do tempo de absorção do modelo, nesse caso, levando-se em conta o tempo associado à transição *capturar_foto* (4.406). A métrica de consumo de energia é obtida através da soma da energia estimada de cada função da aplicação móvel. Por fim, a métrica da disponibilidade apresenta a probabilidade do *smartphone* está operacional juntamente com um dos seguintes conjuntos de elementos: (i) MCC e Sinal da Torre 3G, ou (ii) o *access point* e a *cloudlet*.

Tabela 5.7: Equações usadas no TimeNET para obtenção das métricas do estudo de caso.

Métrica	**Função**
Tempo médio de execução	$1/((P\{\#POn > 0\}) * (1/4.105))$
Consumo de energia	$((P\{\#POn > 0\} * 1/4.406) * 94.56 * 4.953) + ((P\{\#foto\_capturada > 0\} * 1/0.335) * 65.462 * 4.953) + ((P\{\#paraWlan > 0\} * 1/0.22) * 7.86 * 4.953) + ((P\{\#para3G > 0\} * 1/4.535) * 3308.17 * 4.953)$
Disponibilidade	$P\{\#fone\_operacional > 0 \ AND \ ((\#MCC\_operacional > 0 \ AND \ \#3G\_operacional > 0) \ OR \ (\#Acp\_operacional > 0 \ AND \ \#cloudlet\_operacional > 0))\}$

5.4 Estudo de caso 3: *Upload* de arquivos não comprimidos

O intuito da abordagem apresentada neste trabalho é facilitar a modelagem de uma aplicação para melhor construí-la, analisando variações também no comportamento da aplicação. Dessa forma, nesse estudo de caso, a funcionalidade de compressão das imagens capturadas pela aplicação foi retirada da modelagem com o intuito de verificar se tal mudança pode representar um ganho de consumo de energia e tempo de execução significativo ou não. Nas análises foram utilizados os parâmetros mostrados na Tabela 5.8 para *upload* de imagens não comprimidas. Os valores desses parâmetros foram obtidos através de experimentos, seguindo a metodologia explicada anteriormente na Seção 4.2.1. Além disso, análises foram executas nas DSPNs do cenário base adotado (Seção 4.3), dessa vez, sem considerar as restrições de falhas (apresentadas na Seção 4.2.1), para ser possível obter métricas de disponibilidade. Durante o processo de validação (Seção 5.1) as restrições de falha dos elementos foram consideradas para representar as mesmas condições dos experimentos.

O cenário da arquitetura base mostrou variações das métricas para o *upload* de imagens não comprimidas (ver Figura 5.10). Como é possível observar nos gráficos, o tempo médio

Tabela 5.8: Parâmetros para *upload* de imagem (.jpg) de 1MB.

Operação	**Tempo médio de execução (s)**	**Consumo de médio energia (*j*)**
Upload via WLAN	0,266	9,073
Upload via WiFi	9,938	605,582
Upload via 3G	4,623	3780,83

de execução da aplicação foi reduzido em 0,715 segundos. O consumo de energia também sofreu uma redução para 103,629 joules, apresentando uma redução de 95,283 joules. Isto pode ser explicado devido ao aumento da disponibilidade da aplicação que é maior para *upload* de arquivos .jpg, se comparados com arquivos .zip. O valor da disponibilidade sofreu elevação, chegando a uma disponibilidade de 98,463%.

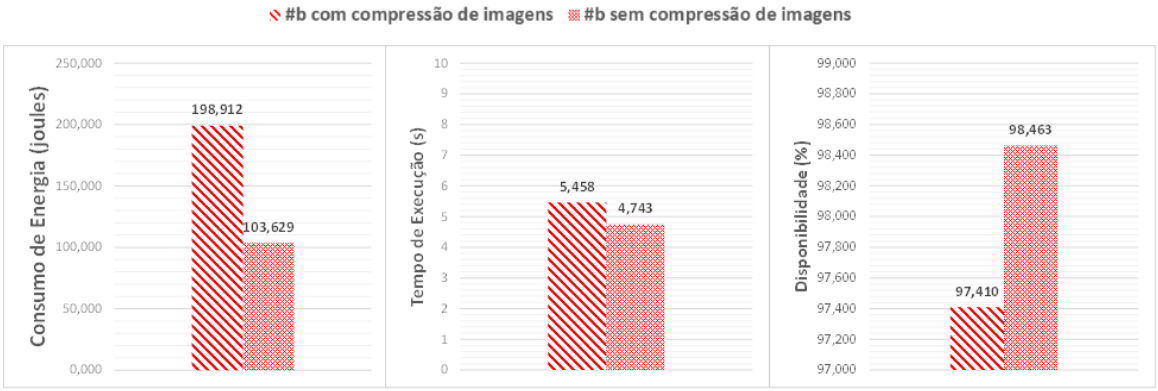


Figura 5.10: Comparação da arquitetura base com compressão de imagens e sem compressão de imagens.

Para o cenário de *uploads* via múltiplas interfaces de rede de arquivos .jpg, foi observada uma pequena perda de qualidade das métricas analisadas, conforme apresentado na Figura 5.11. O tempo de execução médio chegou a 14,269 segundos, representando um aumento de 3,125 segundos no tempo médio de execução. Além disso, o consumo de energia chegou a 764,353 joules apresentando um aumento de 175,231 joules. Já a disponibilidade do sistema apresentou uma leve queda. A perda na qualidade das métricas está relacionada com o aumento dos parâmetros de tempo para o *upload* via rede WiFi de arquivos não comprimidos e também ao aumento do consumo de energia. Mesmo retirando uma operação da aplicação móvel (a compressão de imagens), as métricas se mostram piores. Dessa forma, para este cenário, é melhor que os projetistas utilizem a função de compressão de arquivos, pois para *upload* de arquivos comprimidos (.zip) os valores das métricas são melhores.

O terceiro cenário, utilizando a arquitetura de *cloudlet* também mostrou pequenas alterações nos resultados das métricas (ver Figura 5.12). O tempo médio de execução da aplicação apresentou leve queda de 0,238 segundos. O consumo médio de energia também sofreu leve queda em 49,615 joules. A disponibilidade apresentou um leve aumento, chegando a presentar

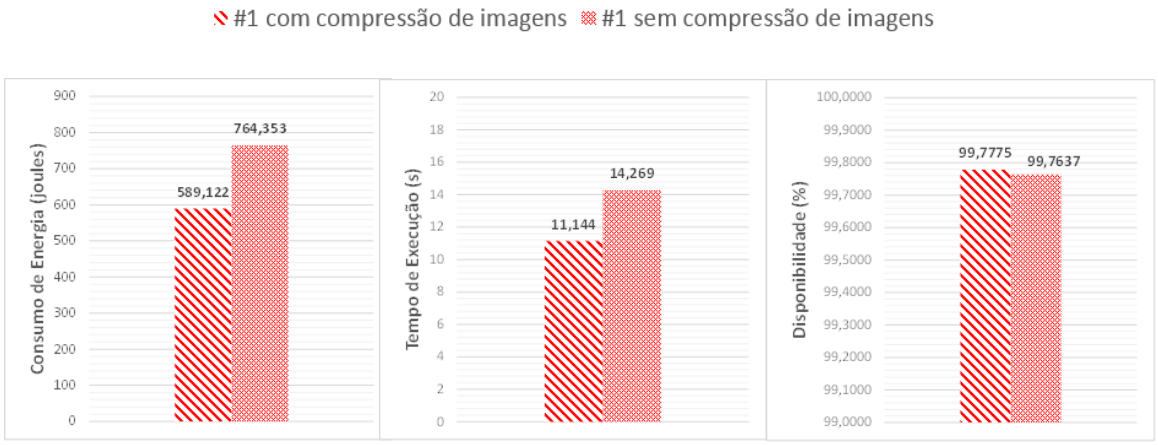


Figura 5.11: Comparação da arquitetura com *upload* via múltiplas interfaces de rede, com compressão de imagens e sem compressão de imagens.

três 9's, com a disponibilidade de 99,933%. Note que as diferenças nas métricas foram muito sutis, se comparado aos demais cenários.

É possível notar na comparação entre todos os cenários, que o consumo de energia acompanha a variação do tempo médio de execução da aplicação. Quando há uma redução no tempo de execução, o consumo de energia é reduzido, e quando há um aumento no tempo de execução, o consumo de energia também aumenta. As comparações mostraram que o *upload* de arquivos comprimidos (.zip) através da aplicação testada só é mais eficiente que o *upload* de arquivos não comprimidos (.jpg) no cenário do estudo de caso 1 (ver Figura 5.11).

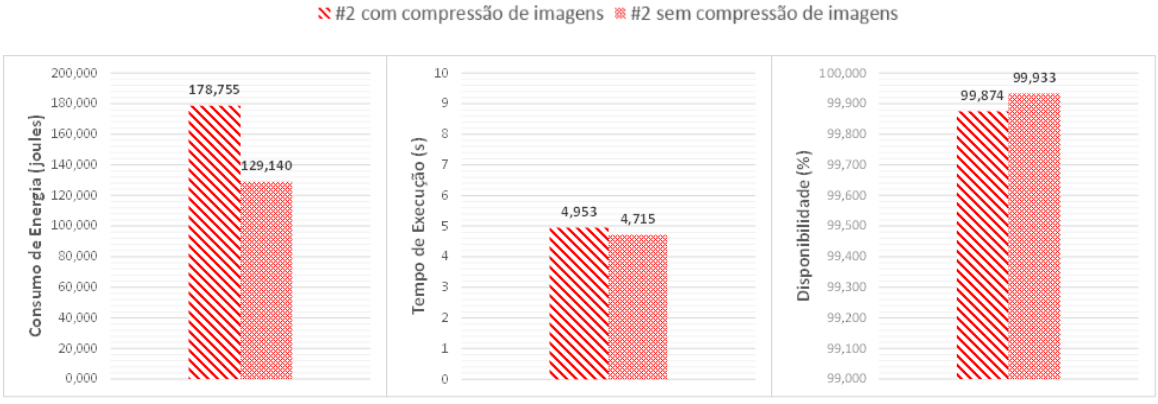


Figura 5.12: Comparação da arquitetura de *cloudlet* utilizando *upload* com compressão de imagens e sem compressão de imagens.

5.5 Comparação entre as arquiteturas utilizadas

Nesta seção, são comparadas as arquiteturas utilizadas para o desenvolvimento da aplicação móvel utilizada nos estudos de casos mostrados anteriormente. A Tabela 5.9 mostra as métricas estimadas pelas análises estocásticas realizadas nos modelos dos três cenários avaliados nos estudos de caso para arquivos comprimidos, onde o identificador #b equivale ao cenário da arquitetura base, #1 para o *upload* através de múltiplas interfaces de rede e #2 para o *upload* utilizando a arquitetura de *cloudlet*.

Tabela 5.9: Métricas estimadas pelos modelos nas arquiteturas utilizadas.

Cenário	Tempo médio de execução (s)	Consumo médio de energia (*j*)	Disponibilidade (%)
#b	5,458	198,912	97,401
#1	11,144	589,122	99,777
#2	4,953	178,755	99,874

É possível notar que o cenário base possui a menor disponibilidade entre os três cenários. A baixa disponibilidade pode explicar a diferença dos resultados de +0,486 segundos no tempo de execução e +41,054 no consumo médio de energia da aplicação entre as análises do cenário base com as restrições (na Seção 5.1) e sem as restrições. Mesmo dessa forma, o cenário da arquitetura base (para arquivos comprimidos) apresenta o segundo menor consumo de energia. Isto pode ser explicado através do uso de um único tipo de conexão de rede (WLAN) para comunicação entre o dispositivo móvel e a MCC.

O cenário com *upload* através de múltiplas interfaces de rede apresenta o maior tempo médio de execução e o maior consumo de energia para a aplicação. Isto acontece devido a latência da rede WiFi para realizar *uploads* através da Internet, uma vez que é comum a taxa de *upload* de redes domésticas serem um pouco mais baixas. Porém, os *uploads* realizados via rede 3G também contribuem para o aumento do consumo médio de energia da aplicação. Contudo, a rede 3G também auxilia na disponibilidade da aplicação, fazendo com que o serviço apresente a segunda melhor taxa de disponibilidade de 99,777%.

A utilização da arquitetura de *cloudlet* fez com que este cenário apresentasse a maior disponibilidade dentre os três analisados. Este cenário se mostra melhor em relação ao demais. Possui o menor tempo médio de execução da aplicação, a maior disponibilidade e o melhor consumo médio de energia. Contudo, é necessário ficar atento aos valores de MTTF e MTTR do *access point*, pois se a disponibilidade do *access point* for menor, pode implicar num uso maior da rede 3G, o que pode acarretar num consumo de energia maior da aplicação.

5.6 Considerações Finais

Neste capítulo foram discutidos estudos de caso que se utilizaram da abordagem apresentada no trabalho proposto para a realização de análises. Além disso, foi mostrado o processo de

validação para os modelos DSPNs gerados através das regas de mapeamento dos diagramas da SysML. Os estudos de caso mostraram variações de infraestrutura que poderiam ser utilizadas por uma aplicação, bem como mudanças comportamentais da própria aplicação testada. Os estudos de caso mostraram que a utilização de uma infraestrutura baseada na arquitetura de *cloudlets* pode aumentar a disponibilidade de aplicações móveis e tornar a aplicação mais eficiente quanto ao tempo de execução e consumo de energia para operações que se utilizem de conexões de rede.

6

Conclusões e Trabalhos Futuros

People tend to give up. If you have persistence, you will come out ahead of most people. More importantly, you will learn. When you do something, you might fail. But that's not because you're a failure. It's because you have not learnt enough. Do it differently each time. One day, you will do it right. Failure is your friend.

—JORDAN BELFORT, THE WOLF OF WALL STREET

A propagação dos *smartphones*, *tablets* e dispositivos inteligentes em geral vem mudando o mercado da computação móvel. O número de aplicações móveis nas duas principais lojas virtuais de aplicativos (*Google Play e Apple Store*) já somam quase 3 milhões (STATISTA, 2015). Isso mostra a crescente demanda por soluções móveis para atender as novas necessidades dos usuários. Contudo, muitos desses aplicativos são desenvolvidos sem um bom planejamento no que se refere a uma boa eficiência energética. Apesar dos projetistas serem incentivados a criarem aplicações energeticamente eficientes, a principal dificuldade é determinar quais são as melhores escolhas arquiteturais a serem feitas no projeto do software (ZHANG et al., 2010).

Nesse sentido, este trabalho apresentou uma abordagem que utiliza modelos (semiformais e estocásticos), para realizar avaliação de métricas de desempenho (tempo de execução) consumo de energia e disponibilidade de aplicações móveis. A abordagem permite auxiliar projetistas de sistemas que possuem pouco conhecimento em modelagem estocástica a projetar e analisar o comportamento de aplicações móveis a partir da construção de modelos semiformais em SysML com anotações do *profile* MARTE. A abordagem proposta consiste em quatro passos: primeiramente o projetista deve definir o cenário para estudo. Depois fazer a modelagem deste cenário em diagramas da SysML anotados com MARTE. Posteriormente, DSPNs são obtidas a partir dos diagramas SysML criados. Por fim, as métricas de interesse são calculadas. As métricas obtidas a partir da realização de análises estocásticas nos modelos DSPNs gerados podem ajudar os projetistas nas tomadas de decisões arquiteturais do projeto da aplicação móvel, tornando possível a criação de aplicações mais eficientes, tanto com relação ao desempenho

quanto ao consumo de energia.

Experimentos foram realizados para se obter dados reais de desempenho (tempo de execução) e consumo de energia das aplicações móveis. A metodologia adotada para realização dos experimentos foi detalhada. Primeiramente, foram criados protótipos da aplicação, separando as funcionalidades em diferentes processos. Posteriormente, um número pequeno de experimentos iniciais foi realizado para cada funcionalidade. Após isso, foi calculado o número mínimo de experimentos necessários para cada funcionalidade, de forma que os valores apresentados estivessem dentro do intervalo de confiança adotado. Por fim, a quantidade mínima de experimentos foi realizada e as médias de cada um dos valores foi apresentada. Estes valores serviram como parâmetros para validação dos modelos obtidos.

Para demostrar a aplicabilidade da abordagem apresentada, estudos de casos foram realizados. No primeiro, foram utilizadas múltiplas interfaces de rede para realização do *upload* de imagens comprimidas para servidores alocados em uma MCC. No segundo estudo de caso, além das múltiplas interfaces de rede, foi utilizada uma infraestrutura de *cloudlet*. No terceiro, foram comparados os resultados obtidos nos dois estudos de casos anteriores e os resultados do cenário base de estudo. Também foi realizada uma variação de cenário, retirando-se a função de compressão de imagens da aplicação móvel, a fim de verificar se traria benefícios de desempenho e diminuição do consumo de energia da aplicação.

Os resultados destes estudos de caso mostraram diferentes valores para as métricas, de acordo com cada cenário em questão. Apresentando, dessa forma, as diferentes possibilidades que projetistas podem ter para realização de análises, sendo possível variar tanto a infraestrutura que será utilizada pela aplicação, como o próprio software. O melhor resultado nas análises para o tempo de execução da aplicação foi notado utilizando a arquitetura de *cloudlet*. Já o para o consumo médio de energia da aplicação, foi notado o melhor resultado nas análises utilizando o cenário base. Também foi percebido um aumento no consumo médio de energia da aplicação quando o tempo médio de execução da aplicação também aumenta, mostrando a relação entre essas duas métricas. Esse é só um exemplo prático de como tipo de informação é valiosa na hora de se planejar aplicações. Assim, elas podem ser construídas da melhor forma possível para suprir as necessidades dos usuários, respeitando os recursos disponíveis e tentando ser o mais eficiente possível, tanto em questões energéticas quanto de desempenho.

6.1 Contribuições

Nesta seção são apresentadas as contribuições deste trabalho:

- A abordagem desenvolvida é uma contribuição que pode auxiliar projetistas na análise de métricas de desempenho, consumo de energia e disponibilidade ainda durante a fase de planejamento da aplicação móvel.

- A definição de regras para mapeamento de diagramas SysML em DSPNs. O mapeamento auxilia na realização de análises de métricas de desempenho e consumo de energia. Outros trabalhos também definem regras de mapeamentos de modelos semi-formais para modelos formais, contudo não as utilizam para análises de aplicações móveis;

- O método adotado na execução de experimentos para medição de consumo de energia. Este método faz uso de fórmulas matemáticas para que os valores dos experimentos se apresentem dentro de uma margem de erro aceitável, dado um determinado intervalo de confiança;

- Aceitação de artigo em conferência internacional (Qualis B2):

  Júlio Mendonça, Ricardo Lima, Ermeson Andrade and Gustavo Callou. Assessing Performance and Energy Consumption in Mobile Applications. In: Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2015). October 9-12, 2015 - Hong Kong, China.

6.2 Limitações

As limitações do trabalho são:

- A criação e atribuição das funções de guarda são feitas de forma manual. Portanto, podem acontecer erros por parte dos projetistas quando essa tarefa for realizada;

- Apenas as métricas de desempenho e consumo de energia foram validadas. Sendo as métricas de disponibilidade apresentadas apenas por estimativas do resultado das análises dos modelos utilizados;

- Os estudos de casos realizados apresentam uma quantidade baixa de funcionalidades e complexidade se comparados a outras aplicações mais populares nas lojas de aplicativos.

6.3 Trabalhos Futuros

A seguir são listados os trabalhos futuros que serão realizados:

- Realizar experimentos de injeção de falhas para validação das métricas de disponibilidade apresentadas pelos modelos. Esse tipo de experimento pode fornecer métricas mais precisas de MTTF e MTTR para a infraestrutura que é utilizada nos experimentos que envolvem a aplicação móvel;

- Realizar mais testes com a abordagem desenvolvida, levando-se em conta outros métodos de obtenção de parâmetros e outros tipos de aplicações. Esses testes podem provar a generalização da abordagem, independentemente de como são obtidos os parâmetros para uso nos modelos, bem como o contexto no qual a aplicação está inserida.

- Acrescentar à ferramenta OpenMADS (ANDRADE et al., 2013) as regras de mapeamento definidas neste trabalho para análise de desempenho e consumo de energia.

- Realizar testes utilizando *smartphones* que permitem o uso do 3G/4G e WiFi juntos para *download/upload* de arquivos. Esse tipo de abordagem pode impactar diretamente nas métricas de desempenho e consumo de energia do dispositivo móvel.

Referências

AALST, W. van der; HEE, K. van; REIJERS, H. Analysis of discrete-time stochastic petri nets. **Statistica Neerlandica**, [S.l.], v.54, n.2, p.237–255, 2000.

ABOLFAZLI, S. et al. Cloud-Based Augmentation for Mobile Devices: motivation, taxonomies, and open challenges. **Communications Surveys Tutorials, IEEE**, [S.l.], v.16, n.1, p.337–368, First 2014.

ANDRADE, E. **Modelagem e análise de especificações de sistemas embarcados de tempo-real críticos com restrições de energia**. 2009. 112p. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco, Recife, Pernambuco.

ANDRADE, E. C. **Modelagem e Análise de Mecanismos de Tratamento de Interrupções em Infraestruturas Computacionais dos Sistemas Distribuídos**. 2014. 138p. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Pernambuco - UFPE, Recife, Pernambuco.

ANDRADE, E. et al. OpenMADS: an open source tool for modeling and analysis of distributed systems. In: BITSCH, F.; GUIOCHET, J.; KAâNICHE, M. (Ed.). **Computer Safety, Reliability, and Security**. [S.l.]: Springer Berlin Heidelberg, 2013. p.277–284. (Lecture Notes in Computer Science, v.8153).

API Evangelist. **Backend as a Service (BaaS)**. [Online]. Disponível em: `http://baas.apievangelist.com/`. Acessado em: 20/03/2015.

ARMBRUST, M. et al. A View of Cloud Computing. **Commun. ACM**, New York, NY, USA, v.53, n.4, p.50–58, Apr. 2010.

BADGER, L. et al. Cloud Computing Synopsis and Recommendations. In: **Anais...** [S.l.: s.n.], 2012. [Online]. Disponível em: `http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf` Acessado em: 12/01/2015.

BALASUBRAMANIAN, N.; BALASUBRAMANIAN, A.; VENKATARAMANI, A. Energy Consumption in Mobile Phones: a measurement study and implications for network applications. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT CONFERENCE, 9., New York, NY, USA. **Proceedings...** ACM, 2009. p.280–293. (IMC '09).

CALLOU, G. **Energy Consumption and Execution Time Estimation of Embedded System Applications**. 2009. 111p. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco, Recife, Pernambuco.

COMSCORE. **The U.S. Mobile App Report**. [S.l.: s.n.], 2014. [Online]. Disponível em: `http://www.comscore.com/layout/set/popup/content/download/26291/1346569/version/1/file/The+US+Mobile+App+Report.pdf`. Acessado em: 05/05/2015.

COOPER, T.; FARRELL, R. Value-Chain Engineering of a Tower-Top Cellular Base Station System. In: VEHICULAR TECHNOLOGY CONFERENCE, 2007. VTC2007-SPRING. IEEE 65TH. **Anais...** [S.l.: s.n.], 2007. p.3184–3188.

DANTAS, J. et al. An availability model for eucalyptus platform: an analysis of warm-standy replication mechanism. In: SYSTEMS, MAN, AND CYBERNETICS (SMC), IEEE INTERNATIONAL CONFERENCE. **Anais...** [S.l.: s.n.], 2012. p.1664–1669.

DINH, H. T. et al. A survey of mobile cloud computing: architecture, applications, and approaches. **Wireless Communications and Mobile Computing**, [S.l.], v.13, n.18, p.1587–1611, 2011.

DOLEZAL, J.; BECVAR, Z. Methodology and tool for energy consumption modeling of mobile devices. In: WIRELESS COMMUNICATIONS AND NETWORKING CONFERENCE WORKSHOPS (WCNCW), 2014 IEEE. **Anais...** [S.l.: s.n.], 2014. p.34–39.

EMARKETER. **2 Billion Consumers Worldwide to Get Smart(phones) by 2016**. [Online]. Disponível em: `http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694`. Acessado em: 10/01/2015.

FLINN, J.; SATYANARAYANAN, M. Energy-aware Adaptation for Mobile Applications. **SIGOPS Oper. Syst. Rev.**, New York, NY, USA, v.33, n.5, p.48–63, Dec. 1999.

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **A Practical Guide to SysML**: systems modeling language. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

GERMAN, R. **Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets**. New York, NY, USA: John Wiley & Sons, Inc., 2000.

GUEDES, G. T. A. **UML - Uma abordagem prática**. [S.l.]: Novatec Editora, 2008.

HARJULA, E.; KASSINEN, O.; YLIANTTILA, M. Energy consumption model for mobile devices in 3G and WLAN networks. In: CONSUMER COMMUNICATIONS AND NETWORKING CONFERENCE (CCNC), 2012 IEEE. **Anais...** [S.l.: s.n.], 2012. p.532–537.

HAUSE, M. The SysML Modelling Language. In: FIFTEENTH EUROPEAN SYSTEMS ENGINEERING CONFERENCE. **Anais...** [S.l.: s.n.], 2006.

INDEX, C. V. N. **Cisco Visual Networking Index**: global mobile data traffic forecast update, 2014–2019. [S.l.: s.n.], 2015. [Online]. Disponível em: `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf`. Acessado em: 19/03/2015.

JAIN, R. **The Art of Computer Systems Performance Analysis**. New York, NY, USA: John Wiley & Sons, 1991.

JANOUSEK, V. **Modelling Objects by Petri Nets**. 1998. Tese (Doutorado em Ciência da Computação) — Brno University of Technology.

JENSEN, K. Coloured petri nets: a high level language for system design and analysis. In: ROZENBERG, G. (Ed.). **Advances in Petri Nets 1990**. [S.l.]: Springer Berlin Heidelberg, 1991. p.342–416. (Lecture Notes in Computer Science, v.483).

JENSEN, K. **Coloured Petri Nets**: basic concepts, analysis methods and practical use, vol. 2. London, UK, UK: Springer-Verlag, 1995.

JOHN, L. K.; EECKHOUT, L. **Performance Evaluation and Benchmarking**. [S.l.]: CRC Press, 2006.

KIM, D. S.; MACHIDA, F.; TRIVEDI, K. Availability Modeling and Analysis of a Virtualized System. In: DEPENDABLE COMPUTING, 2009. PRDC '09. 15TH IEEE PACIFIC RIM INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2009. p.365–371.

KOVACHEV, D.; CAO, Y.; KLAMMA, R. Mobile Cloud Computing: a comparison of application models. **Computing Research Repository, CoRR**, [S.l.], v.abs/1009.3088, 2010.

KUMAR, K. et al. A Survey of Computation Offloading for Mobile Systems. **Mob. Netw. Appl.**, Secaucus, NJ, USA, v.18, n.1, p.129–140, Feb. 2013.

MACIEL, P. R. M.; LINS, R. D.; CUNHA, P. R. **Introduçao às redes de Petri e aplicações**. [S.l.]: UNICAMP - Instituto de Computação, 1996.

MARSAN, M. A. Stochastic petri nets: an elementary introduction. **Advances in Petri Nets 1989**, [S.l.], p.1–29, 1990.

MARSAN, M.; CHIOLA, G. On Petri nets with deterministic and exponentially distributed firing times. In: **Advances in Petri Nets 1987**. [S.l.: s.n.], 1987. v.266, p.132–145.

MERLIN, P.; FARBER, D. J. Recoverability of Communication Protocols–Implications of a Theoretical Study. **Communications, IEEE Transactions on**, [S.l.], v.24, n.9, p.1036–1043, Sep 1976.

MEULEN, R. van der; RIVERA, J. **Gartner Says by 2017, Mobile Users Will Provide Personalized Data Streams to More Than 100 Apps and Services Every Day**. 2014. 1p.

MINITAB. **Minitab**. [Online]. Disponível em: `http://www.minitab.com/pt-br/products/minitab/features/`. Acessado em: 02/05/2015.

MOBILITY, M. **Motorola XT1033 specifications**. [Online]. Disponível em: `http://www.motorola.com.br/Moto-G-da-Motorola/Moto-G-BRPT.html`. Acesso em 10/12/2014.

MURATA, T. Petri Nets: properties, analysis and applications. **Proceedings of the IEEE**, [S.l.], v.vol. 77, n.No. 4, p.541–580, 1989.

OLIVEIRA, D. et al. Availability and Energy Consumption Analysis of Mobile Cloud Environments. In: SYSTEMS, MAN, AND CYBERNETICS (SMC), 2013 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.4086–4091.

OLIVEIRA, D. M. **Análise de Disponibilidade e Consumo Energético em Ambientes de Mobile Cloud Computing**. 2014. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco - UFPE, Recife, Pernambuco.

OMG. Uml 2.0 superstructure specification. In: **Anais...** [S.l.: s.n.], 2004.

OMG. UML Profile for MARTE: modeling and analysis of real-time embedded systems v.1.1. In: **Anais...** [S.l.: s.n.], 2011.

ORACLE. **Class ZipEntry**. [Online]. Disponível em: https://docs.oracle.com/javase/7/docs/api/java/util/zip/ZipEntry.html. Acessado em: 21/04/2014.

ORACLE. **Class System**. [Online]. Disponível em: http://docs.oracle.com/javase/7/docs/api/java/lang/System.html. Acessado em:21/04/2014.

ORACLE. **The History of Java Technology**. [Online]. Disponível em: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html. Acessado em: 30/04/2015.

PEREZ-PALACIN, D.; MIRANDOLA, R.; MERSEGUER, J. QoS and energy management with Petri nets: a self-adaptive framework. **Journal of Systems and Software**, [S.l.], v.85, n.12, p.2796 – 2811, 2012. Self-Adaptive Systems.

PETRI, C. A. **Kommunikation mit Automaten**. 1962. Tese (Doutorado em Ciência da Computação) — Universität Hamburg, Hamburg, Alemanha.

Pew Research Center. **The Smartphone Difference**. [S.l.: s.n.], 2015. [Online]. Disponível em: http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/. Acessado em: 05/05/2015.

PHONES, C. **New research reveals mobile users want phones to have a longer than average battery life**. [Online]. Disponível em: http://catphones.com/news/press-releases/new-research-reveals-mobile-users-want-phones-to-have-a-longer-than-average-battery-life.aspx. Acessado em: 09/05/2014.

PORTILLO, C. A. Gerenciamento eficaz do escopo do projeto. **Project Management Institute - PMI**, [S.l.], 2010. [Online]. Disponível em: https://brasil.pmi.org/brazil/KnowledgeCenter/Articles/~/media/C0A2F2C90BC642368425263603EE4F17.ashx. Acessado em: 12/08/2014.

PRINGLE, T. **Data-as-a-service: the next step in the as-a-service journey**. [S.l.]: OVUM, 2014. [Online]. Disponível em: http://www.oracle.com/us/solutions/cloud/analyst-report-ovum-daas-2245256.pdf. Acessado em: 15/01/2015.

RAHMAN, M.; GAO, J.; TSAI, W.-T. Energy Saving in Mobile Cloud Computing. In: CLOUD ENGINEERING (IC2E), 2013 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.285–291.

ROBINSON, S. **STRATEGY ANALYTICS: cellphone energy gap is widening**. [Online]. Disponível em: http://www.reuters.com/article/2009/04/07/idUS170805+07-Apr-2009+BW20090407. Acessado em: 10/05/2015.

RUDENKO, A. et al. Saving portable computer battery power through remote process execution. **Journal of ACM SIGMOBILE on Mobile Computing and Communications Review.**, [S.l.], v.vol. 2, n.1, January 1998.

SANDS, A.; TSENG, V. **Cell phone comparison study nov 10'**. [Online]. Disponível em: http://www.squaretrade.com/cell-phone-comparison-study-nov-10. Acessado em: 16/03/2015.

SANTACATERINA, M. **7 Statistics You Didn't Know About Cloud Computing**. [Online]. Disponível em: http://blog.nskinc.com/topic/Cloud-computing/IT-Services-Boston/7-Statistics-You-Didn-t-Know-About-Cloud-Computing. Acessado em: 12/06/2015.

SATYANARAYANAN, M. et al. The Case for VM-Based Cloudlets in Mobile Computing. **Pervasive Computing, IEEE**, [S.l.], v.8, n.4, p.14–23, Oct 2009.

SHARMA, R.; KUMAR, S.; TRIVEDI, M. Mobile Cloud Computing: a needed shift from cloud to mobile cloud. In: COMPUTATIONAL INTELLIGENCE AND COMMUNICATION NETWORKS (CICN), 2013 5TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.536–539.

SHERR, I.; TIBKEN, S. **It's 2014. Why is my battery stuck in the '90s?** [Online]. Disponível em: http://www.cnet.com/news/why-batteries-arent-getting-better/. Acessado em: 02/02/2015.

SHETTY, S. **Gartner Says Cloud Computing Will Become the Bulk of New IT Spend by 2016**. [Online]. Disponível em: http://www.gartner.com/newsroom/id/2613015. Acessado em: 10/01/2015.

SHORIN, D.; ZIMMERMANN, A.; MACIEL, P. Transforming UML state machines into stochastic Petri nets for energy consumption estimation of embedded systems. In: SUSTAINABLE INTERNET AND ICT FOR SUSTAINABILITY (SUSTAINIT). **Anais...** [S.l.: s.n.], 2012. p.1–6.

SIFAKIS, J. Use of Petri Nets for Performance Evaluation. In: THIRD INTERNATIONAL SYMPOSIUM ON MEASURING, MODELLING AND EVALUATING COMPUTER SYSTEMS, Amsterdam, The Netherlands, The Netherlands. **Proceedings...** North-Holland Publishing Co., 1977. p.75–93.

SILVA, A. J. d. **Aspectos da modelagem em SYSML ligados à seleção de processador para sistema embutido**. 2006. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Santa Catarina.

SILVA, B. **Mercury Tool**. [Online]. Disponível em: https://sites.google.com/site/mercurytooldownload/. Acessado em: 21/04/2015.

STATISTA. **Number of apps available in leading app stores as of May 2015**. [Online]. Disponível em: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. Acessado em: 10/06/2015.

TAVARES, E. et al. Hard real-time tasks' scheduling considering voltage scaling, precedence and exclusion relations. **Information Processing Letters**, [S.l.], v.108, n.2, p.50 – 59, 2008.

TEAM, S. M. Systems modeling language (sysml) specification. In: NORTH-HOLLAND PUBLISHING CO. **Anais...** [S.l.: s.n.], 2012.

TRIVEDI, K. S. **Probability and Statistics with Reliability, Queuing and Computer Science Applications**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1982.

WALDEN, S. **Cloud computing by the numbers: the rise of the private and hybrid cloud**. [Online]. Disponível em: `http://mashable.com/2015/05/11/cloud-computing-infographic/`. Acessado em: 12/06/2015.

YEAP, G. K. **Practical low power digital VLSI design**. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

ZHANG, L. et al. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: IEEE/ACM/IFIP INTERNATIONAL CONFERENCE ON HARDWARE/SOFTWARE CODESIGN AND SYSTEM SYNTHESIS (CODES+ISSS). **Anais...** [S.l.: s.n.], 2010. p.105–114.

ZIMMERMANN, A.; BODE, S.; HOMMEL, G. Performance and Dependability Evaluation of Manufacturing Systems Using Petri Nets. In: WORKSHOP ON MANUFACTURING SYSTEMS AND PETRI NETS, 17TH INT. CONF. ON APPLICATION AND THEORY OF PETRI NETS (OSAKA, 1. **Anais...** [S.l.: s.n.], 1996. p.235–250.

ZIMMERMANN, A.; KNOKE, M. **TimeNet 4.0**: a software tool for the performability evaluation with stochastic and colored petri nets. user manual. [S.l.]: Techn. Univ., Fak. IV, Elektrotechnik und Informatik, 2007. (Forschungsberichte der Fakultät IV - Elektrotechnik und Informatik // Technische Universität Berlin).