Graduação em Engenharia da Computação

“Calau: Um Ambiente para Modelagem e Análise de Sistemas Embarcados de Tempo-Real”

Por

Marcelo Macêdo Alves

Trabalho de Graduação

Universidade Federal de Pernambuco

Recife, 18 de janeiro de 2013

Universidade Federal de Pernambuco
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

Marcelo Macêdo Alves

"Calau: Um Ambiente para Modelagem e Análise de Sistemas Embarcados de Tempo-Real"

Trabalho de graduação apresentado como requisito parcial à obtenção do grau de Bacharel em Engenharia da Computação, Universidade Federal de Pernambuco, Centro de Informática.

Orientador
Prof. Dr. Paulo Romero Martins Maciel

Co-Orientador
Ermeson Carneiro de Andrade

Recife, 18 de janeiro de 2013

À minha família.

AGRADECIMENTOS

- À Cássia Ró pelo companheirismo e amor demonstrado por todo esse tempo.
- Aos meus pais e irmãos, que são as pessoas em que penso para sempre seguir em frente.
- Em especial a Ermeson, pelo exemplo de profissionalismo e dedicação demonstrados nesses dois anos de trabalho.
- A todos que fazem parte da HiveLog.
- Agradeço ao Prof. Paulo Maciel pelos comentários sempre relevantes e a oportunidade trabalhar no seu grupo de pesquisa.

Vincit qui se vincit.
--DESCONHECIDO

RESUMO

Sistemas Embarcados de Tempo-Real (SETR) geralmente possuem restrições de tempo que precisam ser satisfeitas para o seu correto funcionamento, já que violações das mesmas podem ser catastróficas, e ocasionar, por exemplo, perdas de vidas ou de grandes quantias de recursos financeiros. Além disso, existem sistemas onde energia é outra restrição que também precisa ser satisfeita. Portanto, a análise de especificações é uma tarefa crítica em um projeto de SETR, visto que ela pode fornecer informações importantes na decisão de implementar ou não um SETR em uma dada plataforma de *hardware*.

Linguagens semiformais, tais como SysML (*System Modelling Language*) e UML (*Unified Modelling Language*), são a forma mais comum para modelar especificações de sistemas críticos, devido principalmente à sua notação amigável e intuitiva. Contudo, os modelos semiformais obtidos por essas linguagens não fornecem suporte para avaliação de desempenho das especificações dos sistemas. Em razão disso, faz-se necessário o mapeamento desses modelos em modelos formais. Linguagens formais, tais como redes de Petri, possuem fundamentos matemáticos sólidos, que suportam sua semântica precisa, estimulam a avaliação de desempenho e fornecem suporte para verificações das propriedades qualitativas e análises. Mas ao contrário dos modelos semiformais, esses modelos formais não são intuitivos e requerem um considerável esforço por parte dos projetistas para entenderem a notação utilizada. Assim, é sensato adotar o uso colaborativo dos modelos semiformais e formais.

Neste trabalho, apresentamos Calau, uma ferramenta que suporta o mapeamento automático do diagrama de Estados da SysML em uma Rede de Petri Temporizada com restrições de energia (ETPN), para a estimação do tempo de execução e do consumo de energia dos SETR. Essas estimativas são obtidas nas fases iniciais de desenvolvimento e são utilizadas como um mecanismo de tomada de decisão. As restrições de tempo de execução e anotações de consumo de energia são representadas por anotações do *profile* MARTE (Modelagem e Análise de Sistemas Embarcados de Tempo-Real). Calau também suporta a análise qualitativa do modelo ETPN gerado pelo processo de mapeamento através da integração com a ferramenta INA. Para mostrar a aplicabilidade de Calau, estudos de caso são detalhados.

Palavras-chave: Avaliação de Desempenho, Análise de Requisitos, Consumo de Energia, Rede de Petri Temporizada, Sistemas Embarcados, SysML.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

Os sistemas embarcados estão presentes em praticamente todas as áreas do nosso cotidiano. Utilizamos esses dispositivos diariamente de forma que muitas vezes sequer percebemos que eles estão lá. Caixas eletrônicos, smartphones, leitores de livro digital, geladeiras, micro-ondas, roteadores e filmadoras são alguns exemplos de tais dispositivos. De fato, com o aumento significativo dos sistemas dedicados de controle, a maioria dos dispositivos que usamos no dia-a-dia possui um processador digital, que é responsável por realizar uma tarefa específica [AND09].

Dentre as diversas categorias de sistemas embarcados, há uma cujos resultados além de íntegros, devem ser produzidos dentro de um determinado intervalo de tempo. Aqueles que pertencem a essa categoria são chamados de Sistemas Embarcados de Tempo Real (SETR) [AND09]. Esses sistemas podem ser classificados em duas subcategorias: críticos (*hard*) e não críticos (*soft*). Nos Sistemas de Tempo-Real não críticos, caso as restrições temporais não sejam satisfeitas, poderá ocorrer uma degradação no desempenho do sistema, que poderá ser tolerada. Sistemas desse tipo podem ser encontrados, por exemplo, nos servidores web, nos telefones celulares, nas TVs digitais, nas vídeoconferências, entre outros. Sistemas Embarcados de Tempo-Real críticos são aqueles cuja restrição de tempo deve ser respeitada a todo custo, visto que a violação dela pode ser catastrófica. Exemplos desses sistemas podem ser encontrados em controle automobilístico, equipamentos médicos, aplicações militares, controle aéreo e espacial, centrais nucleares, entre outros [TMSO08, TMS+07].

Avanços tecnológicos têm possibilitado o desenvolvimento de SETR com funcionalidades cada vez mais complexas e sofisticadas, permitindo assim, o surgimento de dispositivos móveis eficientes, precisos e seguros. Esses dispositivos geralmente possuem uma fonte de energia restrita (ex.: bateria), que ao se esgotar, o sistema para de funcionar. Dessa maneira, estudos relativos à conservação/economia de energia tornaram-se extremamente relevantes nos projetos desses sistemas. Com isso, estimativas referentes ao consumo de energia podem fornecer informações importantes aos projetistas tanto relativas ao tempo de vida da bateria, como também, de partes da aplicação que precisam ser otimizadas [AMCNb, AMCNc, TMS+07].

Com o crescimento da heterogeneidade e da complexidade dos SETR, é requerida uma abordagem interdisciplinar no processo de desenvolvimento de tais sistemas, envolvendo as áreas de engenharia de *software*, mecânica, elétrica e eletrônica. Nesse sentido, foi especificada pela OMG *(Object Managment* Group) [OMG89], a linguagem de modelagem semiformal SysML *(System Modelling* Language) [Sys07]. Essa linguagem suporta a especificação, análise, desenho e verificação de sistemas complexos, que podem incluir *hardware, software,* informações, métodos, pessoas e instrumentos. SysML estende UML *(Unified Model Language)* 2.0 [UML05], para aplicação em engenharia de sistemas [Sys07].

A linguagem de modelagem SysML não possui suporte para anotações quantitativas. Essas anotações são extremamente importantes quando se está modelando sistemas embarcados críticos. A ferramenta Calau suporta a combinação de SysML e MARTE (*Modeling and Analysis of Real-Time and Embedded Systems*) [MAR07] para a modelagem dos SETR. MARTE prover facilidades na construção de modelos com informações necessárias à realização de análises específicas, entre elas, estimações do tempo de execução e consumo de energia. Contudo, a combinação de SysML e MARTE carece de semântica formal não sendo possível aplicar diretamente técnicas matemáticas para a avaliação de desempenho e consumo de energia dos SETR.

Dessa maneira, é sensato adotar o uso colaborativo dos modelos semiformais e formais [AND09]. Modelos formais são apoiados por fundamentos matemáticos sólidos, que suportam sua semântica precisa, estimulam a avaliação de desempenho e fornecem suporte para verificações das propriedades qualitativas e análises. Em se tratando de análise de desempenho, os modelos formais mais utilizados são Álgebra Min-Max [HoCU93], Cadeias de Markov [BGdMT98b], teoria das filas [Wal88] e as Redes de Petri temporizadas [MF76]. Em contrapartida, os modelos semiformais são usados largamente para modelagem de requisitos dos SETR, justamente por sua notação amigável e intuitiva.

A integração dos modelos formais e semiformais, bem como, as análises e verificações dos SETR não são tarefas triviais. O tempo para o mapeamento de um modelo semiformal em um modelo formal pode consumir bastante tempo e esforço dos projetistas. Além disso, essa tarefa é propensa a erros por causa da quantidade de estados que podem ser gerados. Assim, é fundamental o desenvolvimento de uma ferramenta que automatize todo esse processo. Essa ferramenta também permitirá uma completa abstração, com relação à utilização dos modelos analíticos usados para as análises.

1.1 OBJETIVOS

O objetivo deste trabalho é implementar uma ferramenta denominada Calau. Ela realiza o mapeamento automático do diagrama de estados da SysML, anotado de acordo com o *profile* MARTE, em uma rede de Petri temporizada com anotações de consumo de energia [Tav06], afim de realizar análises e verificações nas fases iniciais do desenvolvimento dos SETR. Este trabalho é uma extensão do trabalho proposto por [AND09].

A criação dessa ferramenta trará um conjunto de benefícios para o processo de modelagem e avaliação desses sistemas. O tempo do mapeamento é reduzido, uma vez que o modelo ETPN é gerado automaticamente, a partir de especificações descritas em SysML. Devido a este processo automático, também se garante que não ocorreram falhas no processo de conversão do modelo de alto-nível (Diagramas da SysML) para as ETPNs. Esta ferramenta também permitirá uma completa abstração com relação à utilização das ETPN. Isto é, os projetistas poderão obter as estimativas de interesse sem ter conhecimentos específicos sobre as ETPNs.

1.2 TRABALHOS RELACIONADOS

Grande parte dos trabalhos que propõe o mapeamento das linguagem semiformais em modelos analíticos, objetiva a análise quantitativa, mais especificamente os aspectos temporais. Porém, nenhum desses trabalhos enfatiza a avaliação temporal e o consumo de energia, como é suportado pela ferramenta Calau. Outra característica importante sobre esses trabalhos é a adoção de UML para a especificação dos sistemas. No entanto, a linguagem UML foi introduzida para a modelagem de sistemas baseados em *software*. Dessa forma, quando se está modelando SETR, nos quais é necessário possuir uma abordagem interdisciplinar envolvendo as mais diversas áreas (*software*, mecânica, elétrica e eletrônica), SysML é considerada mais adequada, visto que, ela foi desenvolvida para esse propósito [KP00, KP99, TZH05, TZ05, TZ06, MCBD02a].

Pooley and King [KP00, KP99] apresentam como o Diagrama de Colaboração (DC) em combinação com Diagrama de Estados (DE) da UML podem ser sistematicamente transformados em uma *Stochastic Petri Nets* (SPN). Nessa abordagem, os DEs são transformados em modelos SPNs, e estes são combinados através do DC formando um

único modelo. Os estados do DE são mapeados em lugares no modelo SPN e as transições são mapeadas em transições no modelo SPN. Em resumo, uma abordagem intuitiva é apresentada.

Em [TZH05, TZ05, TZ06], Trowitzsch et al. aborda a derivação do DE da UML em uma SPN para avaliação de desempenho de sistemas de tempo-real. Essa abordagem consiste na derivação dos elementos básicos, tais como estados e transições, do DE em representações SPNs correspondentes. Após isso, os fragmentos das SPNs são compostos em um único modelo. O UML SPT [SPT03] é usado como linguagem de especificação para as restrições dos sistemas de tempo-real. Para as análises, o TimeNET (*Timed Net Evaluation*) [GKZH94] é usado. Messeguer et al. [MCBD02a] foca na derivação sistemática do DE da UML em fragmentos de uma LGSPN (*Labeled Generalized Stochastic Petri Net*). Após isso, todos os fragmentos são compostos em um único modelo que represente todo o comportamento do DE. Somente tempos exponencialmente distribuídos são usados nessa abordagem, isto é, tempos determinísticos não são considerados.

Por outro lado, existem alguns estudos que abordam as análises qualitativas [BP01a, BP01b, DdSS, LPK+00]. Em [BP01a, BP01b], Baresi e Pezzé apresentam um conjunto de regras para traduzir uma especificação descrita em UML para uma rede de Petri predicado/transição. Nessa abordagem, os autores propõem a transformação do diagrama de classe em conjunto com o DE em uma rede de Petri predicado/transição. Para cada método de uma classe, é criado um par de lugares de rede de Petri. Um dos lugares indica a requisição do método e o outro indica o retorno do método após sua execução. Além disso, também são identificadas as chamadas externas que cada classe realiza. Para isso, são acrescentados pares de lugares correspondentes às chamadas efetuadas pela classe. Após isso, os estados do DE são mapeados em lugares da Rede de Petri e as transições são mapeadas em transições de redes de Petri. Então, os modelos de estados que representam cada classe são integrados a fim de obter-se uma única rede de Petri. Por fim, simulações, análises de alcançabilidade e verificações do modelo são realizadas.

Similarmente em [DdSS], o autor apresenta uma abordagem parecida ao trabalho de Baresi, no qual é proposta a transformação do diagrama de classe em conjunto com o DE em uma rede de Petri predicado/transição. No entanto, se por um lado Baresi e Pezzé empregam regras de mapeamento para traduzir os DE desenhados pelo projetista em redes de Petri, em [DdSS] os autores sugerem que o comportamento de cada classe seja diretamente descrito pelo projetista por meio de uma rede de Petri. Já em [LPK+00], Lee et al. propõem a derivação de cenários em redes de Petri temporizadas. Nessa abordagem, o diagrama de caso de uso é usado para elicitar os requisitos e criar os cenários. Os cenários são representados pelos diagramas de sequência. Uma vez criados os cenários, então eles são transformados em redes de Petri temporizadas, onde a linha de vida de uma entidade (*lifeline*) é representada por dois lugares e os eventos de chegada e saída na linha de vida são representados por transições. As análises são realizadas de forma a encontrar informações erradas ou esquecidas nos modelos gerados.

Além disso, existem outros estudos relacionados que propõe a transformação de modelos semiformais para modelos formais na qual UML ou SysML não são utilizadas como linguagem de especificação [AMN+05, AMN+06, VCF+06]. Em [AMN+05, AMN+06], Amorim *et al.* propõem um conjunto de passos para o mapeamento do *Live Sequence Chart* (LSC) em uma representação *Coloured Petri Net* (CPN) [Jen92] equivalente. A LSC permite especificar anti-cenários, bem como modelar o que deve ocorrer. O objetivo desse trabalho é realizar análises e verificações das propriedades dos sistemas embarcados. Essas análises e verificações são realizadas através do CPN *Tools*

[VLM+03]. Por fim, Vijaykumar et al. [VCF+06] apresenta o processo de transformação dos *Statechart* [H+87] em uma cadeia de *Markov*.

Alguns destes trabalhos desenvolveram ferramentas que suportam o processo de mapeamento. Em [DSP05], os autores apresentaram uma ferramenta chamada *ArgoPerformance*, que permite a tradução automática de diagramas UML para SPNs, para avaliação de desempenho. Trowitzsch *et al.* [TJZ07] apresentaram uma ferramenta que gera automaticamente modelos SPN a partir de um diagrama de estados. No entanto, não há ferramentas que suportam o mapeamento do diagrama de estados da SysML em uma ETPN [TMSOJ08], a fim de estimar o consumo de energia e tempo de execução.

1.3 ESTRUTURA DO TRABALHO

Capítulo 2 introduz os conceitos teóricos em que este trabalho é baseado, onde serão detalhados os conceitos de Sistemas Embarcados, SysML, MARTE e Rede de Petri. Capítulo 3 descreve a ferramenta Calau e suas funcionalidades. Capítulo 4 apresenta o processo de mapeamento do diagrama de estados em um modelo ETPN. Capítulo 5 apresenta estudos de casos utilizando a ferramenta. Finalmente, Capítulo 6 conclui o trabalho e apresenta os trabalhos futuros.

CAPÍTULO 2

CONCEITOS BÁSICOS

Este capítulo apresenta os principais conceitos em que a ferramenta está baseada. Inicialmente, são introduzidos os Sistemas Embarcados e suas principais características. Em seguida, é apresentado o Diagrama de Estados da SysML. Após isso, o *profile* da UML MARTE é introduzido. Por último, são apresentados alguns conceitos sobre Rede de Petri e uma extensão sua, as Redes de Petri Temporizadas com Restrição de Energia (ETPN), a qual é adotada nesta monografia.

2.1 SISTEMAS EMBARCADOS

Os sistemas computacionais estão presentes em todos os lugares e sob diversas formas. Milhões deles são construídos anualmente para serem usados como computadores pessoais, o que corrobora com o fato deles estarem tão presentes no nosso cotidiano. Outros bilhões deles são construídos a cada ano para outras diversas finalidades. Alguns desses sistemas possuem algumas características em comum, como por exemplo, estão embarcados em sistemas maiores, onde executam funções repetidas e muitas vezes são despercebidos pelo usuário da aplicação. Tais sistemas são chamados de sistemas embarcados. Embora não possuam uma definição precisa, eles são todos aqueles sistemas que não são nem desktop, laptop ou mainframe. Porém, o melhor entendimento sobre esses sistemas se dá ao examinar alguns exemplos e propriedades.

Os sistemas embarcados podem ser encontrados em uma grande variedade de dispositivos eletrônicos, tais como, sistemas aviônicos, sistemas de telemetria, sistemas de antitravamento em freios, controladores de voo, telefones celular, calculadoras de mão, fornos micro-ondas, caixas eletrônicos, videogames, periféricos de computadores. Incorporados a esses sistemas, eles possuem as seguintes características [VH99]:

- **Funcionalidade Específica.** Um sistema embarcado, geralmente, executa um único programa, repetidamente. Em contraste, a um sistema *desktop,* que executa uma variedade de aplicações em *software*.
- **Fortemente Limitado.** Quase todos os sistemas computacionais possuem restrições nas suas especificações de projeto. Porém, ao se tratar de sistemas embarcados, essas restrições são especialmente limitadas. Por exemplo, restrições relacionadas à capacidade da memória, poder de processamento, espaço físico, entre outros.
- **Reativo e Tempo-Real.** Muitos sistemas embarcados devem, continuamente, reagir a mudanças no ambiente do sistema, e devem computar os resultados dessas mudanças dentro de um intervalo de tempo. Caso haja um atraso nessas operações, dependendo da criticidade do sistema embarcado, pode haver consequências graves.

Em contrapartida, cada sistema embarcado possui características próprias, as quais estão relacionadas diretamente à aplicação onde esses sistemas estão inseridos, por exemplo, peso, preço, tamanho, potência dissipada, entre outras. Outra característica importante é a chamada *time-to-market,* que diz respeito ao tempo requerido para projetar e fabricar um sistema a ser vendido para os consumidores. Atualmente, o *time-to-market* tem se tornado bastante exigente em um projeto de sistema embarcado. Assim, introduzir um sistema o mais cedo possível no mercado, pode ser determinante para seu grau de rentabilidade. Com isso, novas metodologias e ferramentas vêm surgindo, para otimizar o

processo de projeto e fabricação de sistemas embarcados, e consequentemente, a redução de seu *time-to-market* [VH99].

2.1.1 Sistemas Embarcados de Tempo-Real

Dentre a grande variedade de sistemas embarcados existentes, há uma classe relacionada a sistemas que são limitados pelo tempo denominados Sistemas Embarcados de Tempo Real (SETR). A corretude da saída desses sistemas depende não somente do próprio resultado, mas também do tempo em que tal resultado foi produzido. De forma semelhante aos sistemas embarcados em geral, esse tipo de sistemas possuem algumas características em comum [Sin96]:

- **Tempo de Resposta Adequado**. Os SETR devem responder a estímulos externos dentro de um intervalo de tempo bem definido, já que a violação dele pode levar a consequências severas.
- **Previsibilidade.** Seu desempenho deve ser o mais previsível possível, ou seja, o seu comportamento funcional e temporal deve ser tão determinístico quanto impõe a sua especificação.
- **Robustez**. Eles devem ser capazes de continuar seu funcionamento mesmo após falhas em sua execução.
- **Precisão.** Devem fornecer resultados o mais preciso possível. Devido às restrições de tempo, vale notar que calcular resultados precisos não é uma tarefa tão simples.
- **Concorrência.** Podem ser distribuídos e executar as tarefas de forma paralela.

2.2 SYSML

A SysML [Sys07] é uma linguagem de modelagem voltada para engenharia de sistemas. Ela foi desenvolvida pela OMG em conjunto com o INCOSE *(International Council on Systems Engineering)* para prover simples, mas poderosas estruturas para modelar uma grande variedade de problemas ligados à engenharia de sistemas. SysML fornece representações gráficas com um fundamento semântico adequado para modelar requisitos, comportamento e estrutura de sistemas, os quais podem ser integrados com outros modelos de análise da engenharia [AMN+10].

Essa linguagem utiliza um subconjunto da UML 2.0 [UML05], chamado UML4SysML, e propõe mais duas estruturas adicionais (diagramas de requisito e paramétrico), o que resulta em um total de nove diagramas. Os diagramas podem ser divididos em quatro partes: requisitos, comportamento, estrutural e paramétrico [Sys07]. A Figura 2.1 apresenta uma visão geral de todos os diagramas da SysML. Esse trabalho adota Diagrama de Estados, que é um diagrama comportamental, por ser adequado para modelagem de SETR [AMN+10].

2.2.1 Diagrama de Estados

O Diagrama de Estados comportamental da SysML (SysML-SM) é usado para representar o comportamento dinâmico do sistema. Os elementos modelados são representados por entidades, e através desse diagrama, é descrito cada estado em que uma entidade possa estar e sua possível mudança, caso um evento ocorra.

O diagrama é composto, basicamente, pelos seguintes elementos [GUE04]:

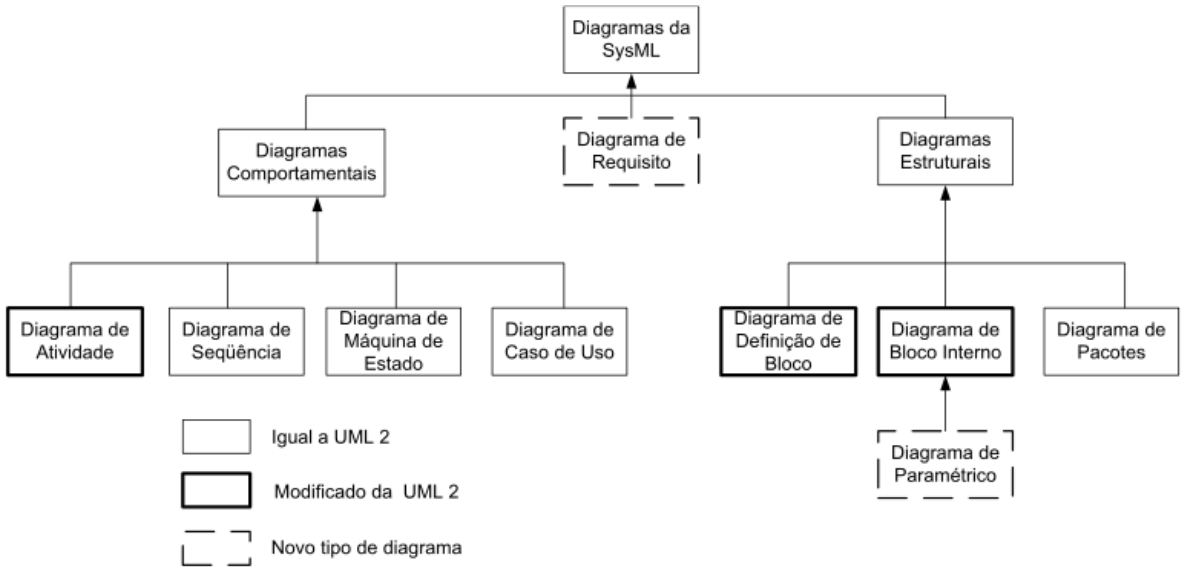

Figura 2.1: Estrutura da SysML

- **Estado.** Representa a situação em que se encontra uma entidade, em um determinado momento durante sua existência em um processo. Há três tipos de estados: estado simples, estado composto e sub-máquina. O Estado Simples é todo aquele estado que não possui nenhum sub-estado, como pode ser visto na Figura 2.2. Estado composto é um estado que contém mais de um estado. Já as sub-máquinas, são estados compostos sequencias ou concorrentes, cujos sub-estados são descritos em outro diagrama.
- **Transição.** Representa um evento capaz de alterar o estado em que uma entidade se encontra. Graficamente, ela é representada por uma seta que liga dois estados (ver Figura 2.2).
- **Estado Inicial.** É um estado cuja função é determinar o início de um Diagrama de Estado ou de um estado composto (ver Figura 2.2).
- **Estado Final.** É um estado cuja função é determinar o fim de um Diagrama de Estado ou de um estado composto (ver Figura 2.2).

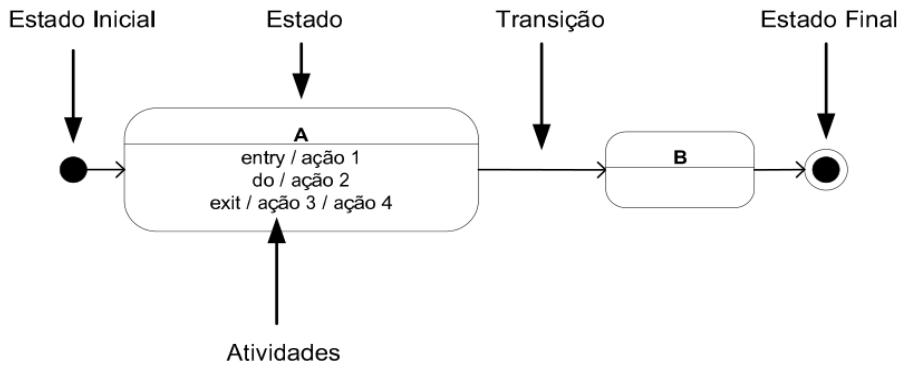

Figura 2.2: Exemplo do diagrama de estados

2.3 MARTE

Embora a SysML seja adequada para a modelagem de SETR, ela não possui suporte para anotações quantitativas, tais como tempo de execução e consumo de energia. Com isso, foi introduzido pela OMG um *profile* da UML 2.0, chamado de MARTE *(Modelling and Analysis of Real-Time and Embedded Systems),* cuja principal função é auxiliar o processo de construção de modelos que possam ser usados para fazer precisões quantitativas relativas às características do sistema, no que diz respeito às propriedades tanto de *hardware* quanto de *software* [MAR07].

As anotações do *profile* MARTE podem ser usadas para diversas finalidades, como por exemplo, escalonamento de tarefas, avaliação de desempenho, entre outras. Para facilitar seu uso, sua especificação foi dividida em unidades de extensão, onde cada uma contém anotações para especificar as características do SETR. Nesse trabalho, foi utilizado a unidade de extensão para Modelagem de Recursos Genéricos (GRM), mais especificamente o estereótipo *ResourceUsage*. O GRM tem como objetivo, oferecer conceitos gerais que são necessários para a modelagem de plataformas de aplicações de tempo-real. Os valores marcados do estereótipo *ResourceUsage*, que são usados nesta monografia, são descritos na Tabela 2.1. Estereótipos são usados para classificar ou introduzir novos elementos na hierarquia de classes do metamodelo, para permitir o incremento da capacidade de modelagem de uma certa linguagem. Valores marcados especificam valores e palavras relativas às características do sistema, sendo normalmente associados a um único estereótipo [MAR07].

A Figura 2.3 demonstra um exemplo de um Diagrama de Estado com restrições de tempo e anotações de energia, ambos especificados pelo *profile* MARTE. Nesse exemplo, pode-se observar que foram usados o estereótipo *ResourceUsage* e os valores marcados *execTime* e *energy.* Esse estereótipo descreve, respectivamente, o tempo de execução e o consumo de energia do estado A, que neste caso são, 10 segundos e 20 *joules*. Os valores marcados são: {*execTime* = (10, s)} e {*energy* = (20, j)}. De forma semelhante, a transição que sai do estado A com destino o estado B, possui um tempo execução e um consumo de energia de, respectivamente, 30 segundos e 40 *joules*. Os valores marcados são: {*execTime* = (30, s)} e {*energy* = (40, j)}.

Tabela 2.1. Valores marcados do estereótipo *ResourceUsage* usados nesta dissertação.

Valores Marcados	Descrição
execTime	Tempo de execução de uma tarefa
energy	Consumo de energia para realização de uma tarefa

2.4 Redes de Petri

Inicialmente proposta na dissertação de doutorado do Dr. Carl Adam Petri, intitulada *Kommunication mit Automaten* (Comunicação com Autômatos), Rede de Petri é, atualmente, usada em diversas áreas tais como Ciências da Computação, Engenharia Elétrica, Administração, Química, entre outras, como uma ferramenta matemática e gráfica que permite modelar sistemas complexos [Pet62].

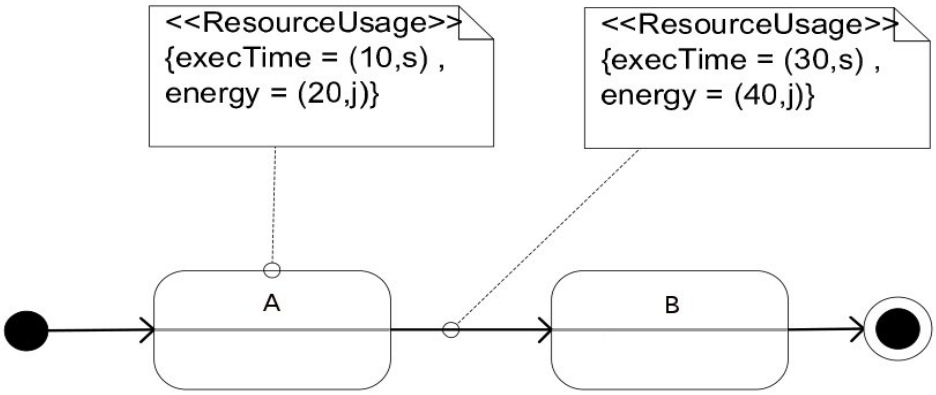

Figura 2.3. Diagrama de Estado com anotações de MARTE.

Devido à necessidade de auxiliar diferentes áreas de sua aplicação, além de prover facilidades de comunicação e transferência de métodos e ferramentas de uma área para outra, diversas variantes de Rede de Petri têm sido desenvolvidas ao longo do tempo, tais como redes temporizadas [MF76], estocásticas [Mar89], alto-nível [Jen91] e orientadas a objetos [Jan98]. Porém, em se tratando apenas do modelo clássico, Redes de Petri são compostas pelos seguintes elementos [MLC96]:

- **Lugares.** Representam os elementos passivos da rede. Têm como principal função armazenar os *tokens*, que são removidos e adicionados na medida em que são disparadas as transições. Graficamente, os lugares são representados por círculos (ver Figura 2.4 (a)).
- **Transições.** Representam os elementos ativos da rede, que são as ações realizadas pelo sistema. Para que uma transição seja habilitada, todas as suas pré-condições precisam ser satisfeitas. Quando isso ocorre, a transição é disparada, o que acarreta na transferência de *tokens* entre lugares. Graficamente, as transições são representadas por traços ou barras. (ver Figura 2.4 (b)).
- **Arcos.** Representam o fluxo dos *tokens* pela rede (ver Figura 2.4 (c)).
- **Tokens.** Representam o estado em que o sistema se encontra em determinado momento (ver Figura 2.4 (d)).

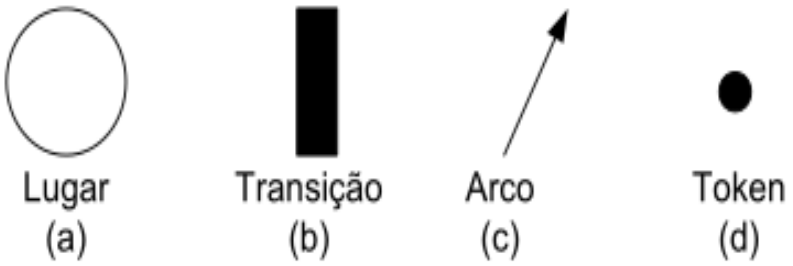

Figura 2.4: Elementos de uma rede de Petri.

Formalmente, Rede de Petri pode ser definida conforme descreve a Definição 2.1.

Definição 2.1 (Rede de Petri) Uma Rede de Petri é uma 5-tupla, *PN = (P, T, F, W, Mo),* tal que:

- *P = {p1, p2, …, pn},* é um conjunto finito de lugares;

- $T = \{t1, t2, \ldots, tn\}$, é um conjunto finito de transições;
- $F \subseteq (P \times T) \cup (T \times P)$, é um conjunto de arcos;
- $W : F \rightarrow \{1, 2, 3, 4, \ldots, n\}$ é a função de distribuição de pesos aos arcos;
- $M0 : P \rightarrow \{0, 1, 2, 3, \ldots\}$ é a marcação inicial, onde $P \cap T = \varnothing$ e $P \cup T \neq \varnothing$.

Na Figura 2.5, é apresentado um gráfico de uma Rede de Petri, na qual é modelado um funcionamento de uma lâmpada comum. Nesse modelo, o estado da lâmpada (aceso ou apagado) é representado pelos lugares e as ações que alteram o estado da lâmpada (ligar ou desligar) são representadas pelas transições. O estado em que se encontra a lâmpada é representado por um *token* inserido no lugar correspondente à situação atual do modelo. Como pode ser vista na Figura 2.5 (a), o estado da lâmpada é *aceso,* e a única transição que poderá ser disparada é *desligar*. Quando essa transição for disparada, o modelo passará, então, do estado *aceso* para o estado *apagado* (ver Figura 2.5 (b)).

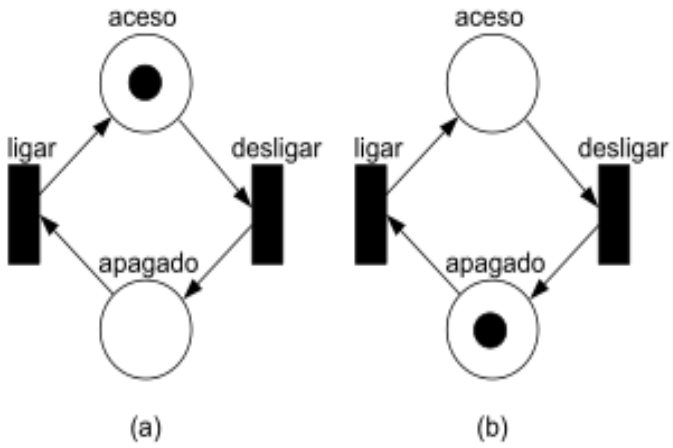

Figura 2.5: Exemplo de uma Rede de petri

Na modelagem de sistemas, lugares e transições podem ter diferentes significados. A Tabela 2.2 mostra possíveis interpretações que podem ser feitas sobre os lugares e as transições. Como exemplo, os lugares podem representar condições e as transições os eventos, sendo que, um evento pode ter várias pré-condições, e também, várias pós-condições [Mu89].

Tabela 2.2: Interpretações para os lugares e transições

Lugares de Entrada	Transições	Lugares de Saída
Pré-condições	Eventos	Pós-condições
Dados de entrada	Passo e computação	Dados de saída
Sinal de entrada	Processamento de sinal	Sinal de saída
Disponibilidade de recursos	Tarefa	Liberação de recursos
Condição	Cláusula lógica	Conclusões
Buffers	Processador	Buffers

Fonte: [Mur89, p.542]

2.4.1 Redes de Petri Temporizadas com Restrição de Energia

Como visto anteriormente, em uma Rede de Petri, uma transição $t \in$ T é disparada, se suas pré-condições forem satisfeitas. Em se tratando de Redes de Petri Temporizadas com Restrição de Energia (ETPN), essas pré-condições estão relacionadas às restrições de tempo. As Redes de Petri Temporizadas foram inicialmente propostas por Merlin e Faber em [MF76]. No entanto, este trabalho adota o modelo no qual a informação temporal é representada por um retardo [TMSO08].

Definição 2.2. (Rede de Petri Temporizada) Uma Rede de Petri Temporizada (TPN) é representada por uma tupla *P= (N, I)*, onde *N* é a Rede de Petri base, e $I: T \rightarrow \mathbb{N} \times \mathbb{N}$ representa as restrições de tempo, onde $I(t) = (EFT(t), LFT(t)) \; \forall \; t \in T$ e $EFT(t) \leq LFT(t)$. Os limites inferior (*EFT*) e superior (*LFT*) representam os tempos máximo e mínimo de disparo de uma transição *t*. *EFT e LFT* denotam *Earliest Firing Time e Latest Firing Time*, respectivamente [MF76].

Definição 2.3. (Rede de Petri Temporizadas com Restrição de Energia) Uma ETPN é representada por *Pe = (P, E)*. *P* representa a Rede de Petri temporizada, e $E : T \rightarrow \mathbb{R}+ \cup \{0\}$ é uma função que associa valores de consumo de energia às transições [MF76].

Definição 2.4 (Transições Habilitadas) Um conjunto de transições habilitadas, na marcação m_j, é denotado por:

$$ET(m_i) = \{ t \in T \mid m_i(p_j) \geq W(p_j, t), \; \forall \; p_j \in P \}$$

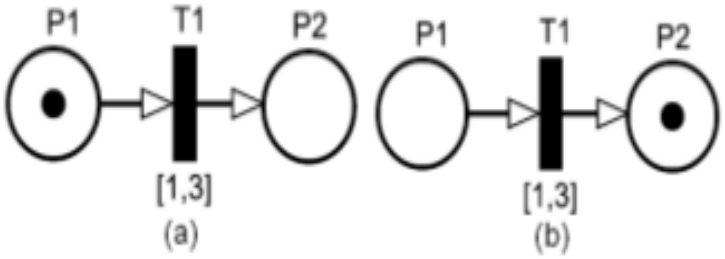

Figura 2.6: Exemplo de uma Rede de Petri Temporizada

Existem dois modos de disparo nas ETPN: *strong* e *weakest*. No modo de disparo *strong*, uma transição habilitada (ver Definição 2.4) não pode ser disparada antes do seu *EFT(t*), e deve ser disparada antes ou no seu *LFT(t).* Por outro lado, o modo de habilitação *weakes*t não força o disparo da transição, ou seja, uma transição habilitada pode ou não disparar. Este trabalho adota o modo de disparo *strong.*

Nas redes de Petri, uma transição $t \in$ T é habilitada, se cada lugar de entrada $p \in$ P contém no mínimo *w(p, t)* tokens. No entanto, nas redes de Petri temporizadas, para que a transição seja disparada, além da marcação, as restrições tempo precisam ser respeitadas. O tempo decorrido, uma vez que a transição esteja habilitada, é representado por um vetor de *clock* $c \in (N \cup \{\#\})^{|T|}$ onde # representa o valor indefinido para as transições. O vetor de clock, por exemplo, da rede da Figura 2.6 (a) contém um elemento: *c(t1) = 0* [AMCNd].

Neste momento, é importante ressaltar a diferença entre disparo de intervalo dinâmico e estático. O intervalo dinâmico de disparo da transição *t*, *ID(t) = [DLB(t),*

DUB(t)], é dinamicamente modificada sempre que a variável *c(t)* do respectivo clock é incrementada, e *t* não dispara. *DLB(t)* é o *Dynamic Lower Bound*, e *DUB(t)* é o *Dynamic Upper Bound*. O intervalo dinâmico de disparo é computado da seguinte forma: *ID(t) = [DLB(t), DU B(t)]*, onde *DLB(t) = max(0, EFT(t) − c(t))*, *DUB(t) = LFT (t) − c(t)*. Sempre que *DLB(t) = 0*, *t* pode disparar, no entando, quando *DUB(t) = 0*, *t* deve disparar, devido ao disparo *strong* adotado. Na Figura 2.6(a), assumindo que *c(t1)* é incrementado em uma unidade de tempo *(c(t1) = 1), ID (t1) = [max(0, 1−1), 3−1] = [0, 2*] [AMCNd].

Definição 2.5. (Estado) Sejam P_e uma ETPN, M ⊆ ℕ o conjunto de marcação de P_e , C ⊆ (ℕ ∪ {#}) o conjunto de vetores de *clock*, e E ⊆ ℝ+ ∪ {0} o conjunto de energia acumulada. O conjunto de estados S de P_e é dado por S ⊆ (M × C × E).

Definição 2.6. (Domínio de disparo) Seja s = (m, c, e) um estado de uma ETPN. O domínio de disparo para uma transição *t* de um estado específico s, é definido pelo seguinte intervalo de tempo:

$$FDs(t) = [DLB(t), min(DUB(t_k))],\ \forall\ t_k \in ET(m).$$

A transição *t* do estado *s* só poderá ser disparada dentro do intervalo expresso por *FDs(t)*. Considerando o modelo ETPN da Figura 2.6(a), no estado inicial *s0 = (m0 = [1, 0], c0 = [0], e0 = 0)*, *t1* é disparavel quando *c(t1) = 1* e deve disparar quando *c(t1) = 3 (FDs(t) = [1, 3]).*

2.5 Considerações Finais

Novas ferramentas e metodologias vêm surgindo em prol da redução do *time-to-market* dos sistemas embarcados. SysML é uma ferramenta adequada para modelagem dos SETR. Com MARTE, tornou-se possível introduzir anotações quantitativas nos modelos da SysML. Já Redes de Petri, pode ser vista como uma ferramenta matemática e gráfica, própria para modelar sistemas complexos.

CAPÍTULO 3

CALAU

Este capítulo tem como objetivo mostrar uma visão geral sobre a ferramenta e suas principais funcionalidades. Inicialmente, será visto a arquitetura técnica da ferramenta, apresentado os métodos abordados para construí-la. Logo depois, será visto o editor SysML-SM e MARTE fornecido pela ferramenta, como também, o editor Rede de Petri. Por último, a exportação do modelo ETPN para a ferramenta *INA* será descrita.

3.1 ARQUITETURA DO SOFTWARE

Calau foi desenvolvida para suportar o processo do mapeamento de SysML-SM para ETPN e sua análise. Esta ferramenta permite que os *designers,* que não têm experiência em modelagem formal, projetem e analisem especificações de SETR quantitativamente em uma plataforma de computação nas nuvens. Em outras palavras, Calau permite que os *designers* estudem e analisem sistemas embarcados desenvolvidos em diferentes plataformas de hardware. Ela pode fornecer informações importantes para o *designer* quanto à possibilidade de implementar uma solução ou não em uma determinada plataforma de hardware. Além disso, o processo de mapeamento do diagrama de estado pode resultar em uma quantidade enorme de estados no modelo ETPN. A ferramenta aborda essa questão, uma vez que tanto o processo de mapeamento e análise são completamente automáticos. Calau pode ser acessada em [AAM12].

Antes de iniciar a implementação, procuramos utilizar uma tecnologia que permitisse que a ferramenta seja carregada em um navegador Web sem a necessidade de instalação. Entre as escolhas, escolhemos uma aplicação applet Java, devido aos seguintes aspectos: (i) Java é uma tecnologia de código aberto, isso leva à continuidade e disponibilidade do tecnologia no futuro, (ii) há uma ampla variadade de ambientes integrados de desenvolvimento (IDEs), tais como: Eclipse [MKF06] e Netbeans [KCC+ 05], (iii) as ferramentas desenvolvidas como Applets Java, podem ser empacotadas em arquivos JAR (Java) e podem ser usados como uma aplicação *desktop* ordinária, (iv) existem várias bibliotecas de código aberto para lidar com a manipulação gráfica em Java, o que poderia tornar a construção do editor de diagramas mais fácil e (v) existe um grande número de ferramentas de código aberto escrita em Java, permitindo a reutilização do código existente.

A tela inicial da ferramenta é mostrada na Figura 3.1. Como pode ser observado, existem alguns menus disponibilizados pela ferramenta que auxiliam a construção dos modelos. No menu *File*, estão disponíveis algumas funcionalidades, que entre outras, permitem a gravação de modelos e abrir modelos pré-gravados. No menu *Diagram*, é possível escolher qual diagrama será mostrado pela ferramenta (Diagrama de Estados ou Redes de Petri). No menu *Edit*, existem algumas outras funcionalidades, como por exemplo, desfazer operações, selecionar todos os componentes do diagrama, etc. No menu *View,* é possível alterar o plano de fundo da ferramenta. No menu *Arrange,* contém um conjunto de funcionalidades que permitem melhor organizar os componentes. Por último, o menu *Help,* contém informações básicas sobre a ferramenta.

As funcionalidades implementadas na ferramenta são brevemente resumidas como:

- **Editor SysML e MARTE -** Os usuários têm a possibilidade de projetar os SETR usando Calau. A ferramenta possui todos os elementos do SysML-SM, tais como: estados, transições, estados final e inicial e estados compostos. Calau também fornece anotações do *profile* MARTE, que podem ser atribuídas aos estados e

transições. Porém, apenas um subconjunto dessas anotações é fornecido por Calau.

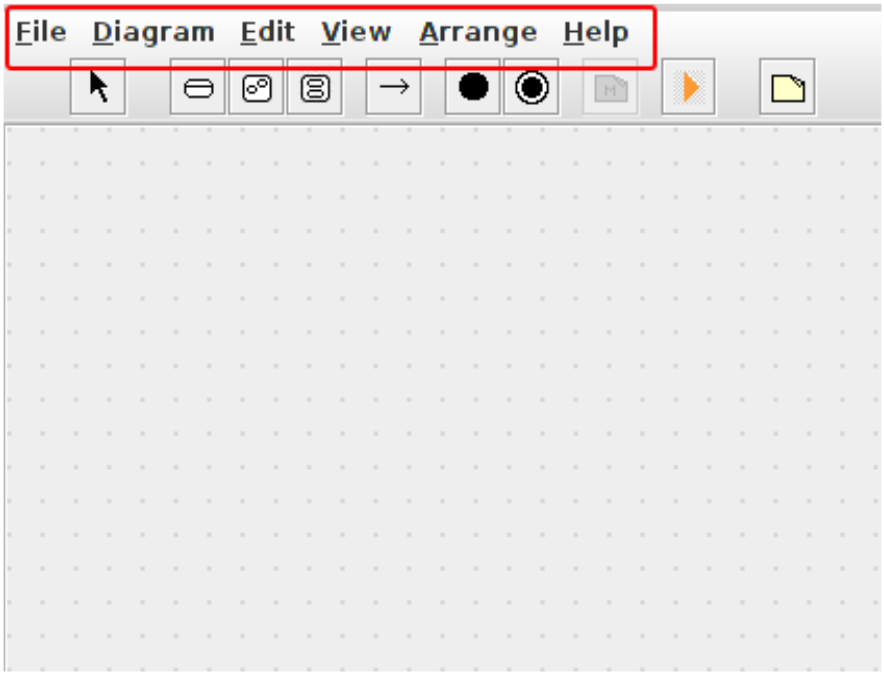

Figura 3.1: Menus da ferramenta

- **Editor de Rede de Petri -** Este editor fornece o modelo ETPN gerado pelo processo de mapeamento. Os usuários podem fazer qualquer modificação no modelo ETPN. Essa funcionalidade permite que os designers considerem diferentes parâmetros para a execução do modelo sem a necessidade de executar o mapeamento novamente.
- **Integração INA -** A fim de verificar um conjunto de propriedades qualitativas, Calau foi integrada à ferramenta INA. Apesar de tais propriedades serem autoexplicativas, é extremamente importante inseri-las dentro do contexto do problema, porque elas podem revelar características importantes do sistema que está sendo modelado.

3.2 EDITOR DO DIAGRAMA DE ESTADOS E MARTE

O editor do diagrama de estados e MARTE da ferramenta são mostrados na Figura 3.2. Como pode ser observado, o editor é composto por três partes principais. A primeira parte compreende os elementos SysML-SM que são estado simples, estado composto, estado concorrente, transição, estado inicial e estado final. A segunda parte contém anotações de MARTE. Apenas o estereótipo (*ResourceUsage*) e os valores marcados (*execTime e energia*) foram incluídos na ferramenta, uma vez que o objetivo é realizar estimativas quanto o tempo de execução e o consumo de energia dos SETR (veja as anotações atribuídas ao estado *Busy*). A terceira parte contém o botão mapeamento. Ao clicar no mesmo, Calau realiza o processo de mapeamento de SysML-SM para o modelo ETPN automaticamente. Detalhes sobre o mapeamento desse processo são apresentados no próximo capítulo. Calau também implementa a funcionalidade de salvar. Ela permite que os usuários criem seus modelos e os guarde para uso futuro ou análise.

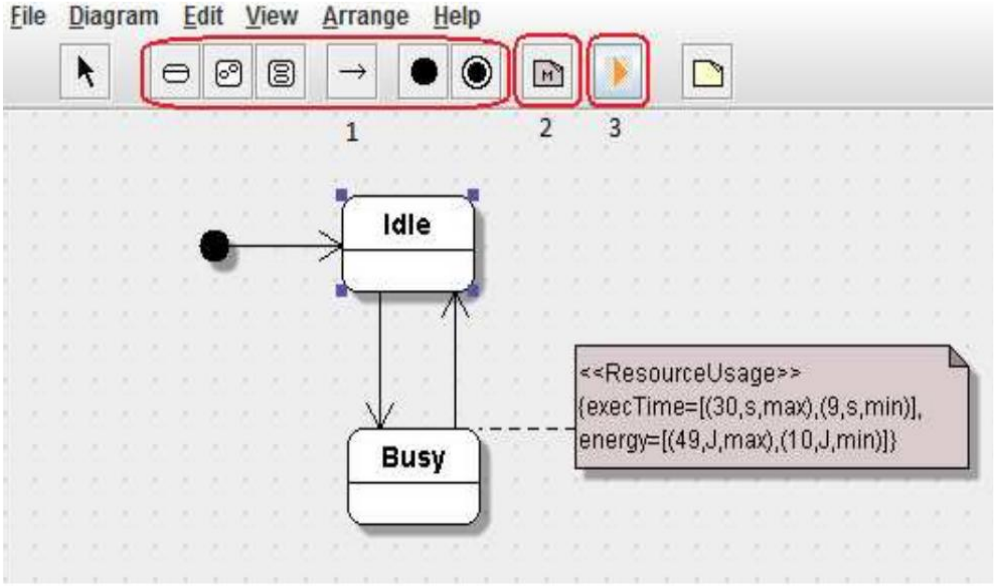

Figura 3.2: Tela do editor SysML e MARTE

3.3 EDITOR DA REDE DE PETRI

A Figura 3.3 apresenta o editor de rede de Petri. De forma semelhante ao editor da SysML e MARTE, ele compreende três partes principais. A primeira parte contém os elementos de rede de Petri. Ou seja, é composto por lugares, transições e arcos. Na segunda parte contém a funcionalidade de exportação para a ferramenta INA. Essa funcionalidade permite que os usuários gerem uma entrada para o INA (ver Seção 3.4). A terceira parte compreende a execução do modelo ETPN. Para isso, primeiro o usuário precisa selecionar um lugar que representa o estado final. Este estado irá identificar onde a execução deverá parar. Depois disso, clicando no botão de execução, Calau calcula as estimativas de tempo de execução e consumo de energia, isto é, a ferramenta realiza a análise quantitativa sobre o modelo.

A análise quantitativa tem como objetivo a avaliação de desempenho e recursos dos sistemas. Nesse trabalho, as estimativas de *BCET (Best Case Execution Time)* e *WCET (Worst Case Execution Time)* [EES+03] são adotadas para os modelos ETPN. No momento em que os *traces* (conjunto ordenado que designa a execução das ações ou eventos) do BCET e WCET são encontrados, então o consumo de energia é encontrado para tais *traces* [EES+03]. A Tabela 3.1 apresenta um resumo das métricas adotadas nessa monografia.

Como pode ser observado na Figura 3.3, cada uma das transições-PN possui um tempo de execução e um consumo de energia associado, onde sempre o primeiro valor associado a transição do modelo é o tempo e o segundo é o consumo de energia. Ambos os valores são estreitos, o que significa que os valores do limite superior e inferior são iguais, a fim de diminuir o espaço de estados do modelo [Dav05].

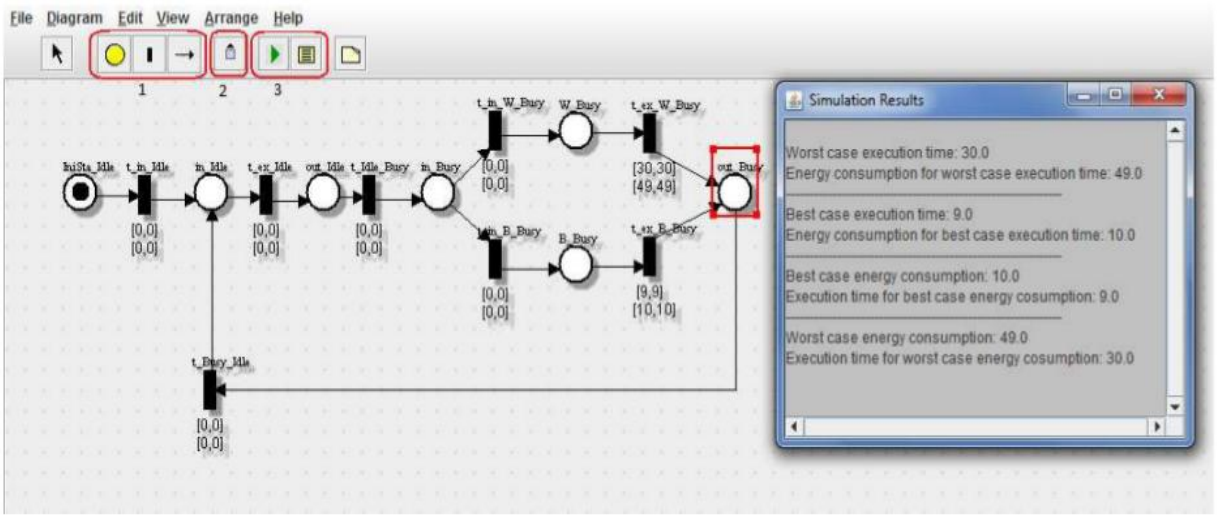

Figura 3.3: Tela do Editor da Rede de Petri

Tabela 3.1: Métricas

Métrica	Descrição
BCET	*Best Case Execution Time*
WCET	*Worst Case Execution Time*
ECBC	*Energy Consumption for Best Case*
ECWC	*Energy Consumption for Worst Case*

3.4 INTEGRAÇÂO COM A FERRAMENTA INA

INA (*Integrated Net Analyser*) é uma ferramenta bem conhecida pela comunidade científica. Ela fornece uma variedade de métodos para análise de Rede de Petri, tais como análise estrutural, análise de acessibilidade e verificação do modelo. Uma das principais vantagens do formalismo da Rede de Petri é a representação gráfica dos modelos. No entanto, a Ferramenta INA não fornece uma interface gráfica. Os usuários têm de transformar a representação gráfica em uma descrição textual manualmente.
Deve-se observar que os modelos podem ser muito grandes e complexos, e assim, a construção do modelo torna-se propensa a falhas.

Calau fornece uma funcionalidade de exportação, na qual permite que os usuários gerem arquivos de entrada para o *INA*. Esses arquivos são usados para realizar análises e verificações do modelo ETPN gerado pelo processo de mapeamento, e com isso, realizar análises de propriedades tal como ausência de *deadlocks* ou reversibilidade. A Figura 3.4 apresenta um exemplo intimamente ligado a essa funcionalidade de exportação.

A Figura 3.5 apresenta o resultado da análise qualitativa feito pelo *INA*, a partir da exportação do modelo ETPN mostrado na Figura 3.3. A saída dessa análise mostra um conjunto de propriedades que são relacionadas ao sistema que está sendo modelado. Mesmo que essas propriedades sejam auto-explicativas, elas podem revelar características importantes do sistema, quando estão inseridas dentro do contexto do problema.

3.5 CONSIDERAÇÕES FINAIS

Neste capítulo, foram apresentadas as principais funcionalidades da ferramenta Calau. Esta ferramenta permite que os SETR sejam modelados através do diagrama de estados da SysML. Esse diagrama, por sua vez, pode ser anotado com informações de tempo de execução e consumo de energia usando o *profile* da UML MARTE. Após isso, é possível realizar o mapeamento automático do diagrama de estados em um modelo analítico, onde o mesmo pode ser editado, analisado ou exportado para a ferramenta INA.

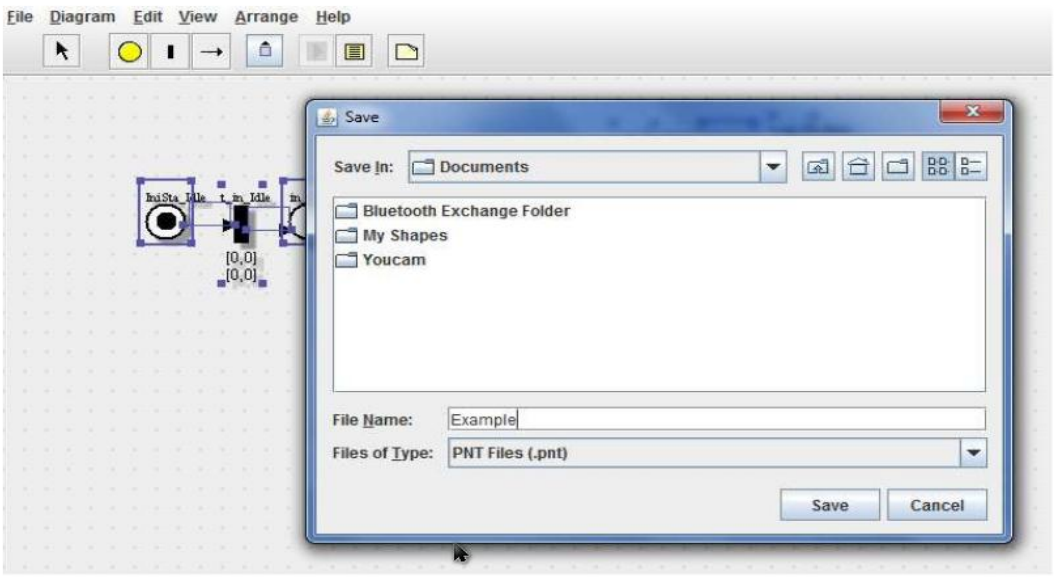

Figura 3.4: Tela da funcionalidade de exportação para *INA*

```
.......................Print the static conflicts? Y/N N
The net is not statically conflict-free.
The net is pure.
The net is ordinary.
The net is homogenous.
The net is conservative.
The net is structurally bounded.
The net is bounded.
The net is subconservative.
The net is a state machine.
The net is extended free choice.
The net is extended simple.
The net is free choice.
The net has places without pre-transition.
The net is not state machine decomposable (SMD).
The net is not strongly connected.
The net is not covered by semipositive T-invariants.
The net is not live.
The net is not live and safe.
The deadlock-trap-property is not valid.
The net is marked.
The net is marked with exactly one token.
The net is not a marked graph.
The net has not a non-blocking multiplicity.
Press a key!
```

Figura 3.5: Resultado da análise qualitativa

CAPÍTULO 4

MAPEAMENTO DO DIAGRAMA DE ESTADOS EM UM MODELO ETPN

Neste capítulo será descrito, em detalhes, o processo de mapeamento do Diagrama de Estados da SysML no modelo ETPN, incluindo as restrições de tempo e consumo de energia. Isto é, serão apresentados os principais elementos que compõe o diagrama e as regras de mapeamento dos mesmos.

4.1 MAPEAMENTO DOS ESTADOS

Um estado representa a situação de uma entidade, na qual pode estar realizando uma atividade ou aguardando um evento. Graficamente, os estados são representados por elipses ou retângulos, com bordas arredondadas, e são compostos por duas ou três divisões (ver Figura 4.1). A primeira delas apresenta o nome do estado. A segunda divisão apresenta as atividades internas (*entry, do* e *exit*). *Entry* representa as ações que são realizadas no momento em que a entidade assume o estado. *Do* representa as ações executadas enquanto a entidade encontra-se no estado. *Exit* representa as ações feitas antes da mudança de estado. A terceira divisão identifica as transições internas. Na seção 4.2, as transições internas serão detalhadas.

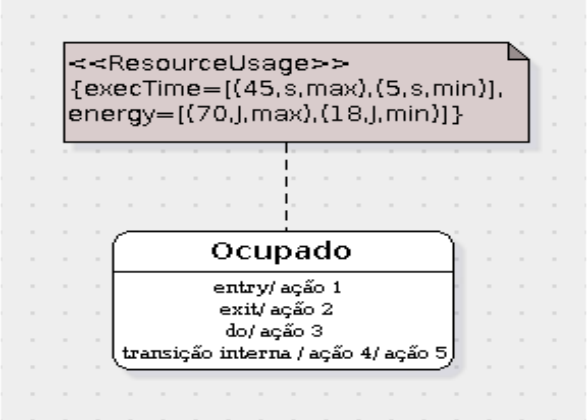

Figura 4.1: Exemplo de um estado simples

A Figura 4.1 mostra um exemplo de um estado simples. Nesse exemplo, o estado *Ocupado* contém atividades internas *(entry, do* e *exit*), uma transição interna e ações que podem ser executadas tanto durante as atividades quanto durante a transição interna. Vale ressaltar que uma ação é algo instantâneo, e assim, não há demanda de tempo em sua execução. Ao contrário dessa, uma atividade necessita de certo tempo para sua execução. O estado também pode possuir restrições de tempo e anotações de consumo de energia (ver Figura 4.1). Nesse exemplo, o tempo de execução para o pior e melhor casos das atividades (*entry*, *do* e *exit*) e da transição interna do estado *Ocupado* são, respectivamente, de 45 e 5 segundos, e o consumo de energia de 70 e 18 *joules*..

A Figura 4.2 apresenta o mapeamento de um estado simples com restrições de tempo e anotações de consumo de energia em um modelo ETPN. São atribuídos às transições-PN *t_ex_W_A* e *t_ex_B_A* valores do limite superior e inferior do tempo de execução no pior e melhor caso, respectivamente. Os valores são estreitos, ou seja, os valores do limite superior e inferior são iguais. Dessa maneira, o tempo de execução máximo e mínimo das atividades *(entry, do* e exit) e da transição interna do estado A são, respectivamente, iguais a [45,45] e [5,5]. As anotações MARTE são usadas para

representar tanto as restrições do tempo de execução quanto as anotações de consumo de energia do estado A. Nesse exemplo, o consumo de energia no melhor e pior caso atribuídos às transições-PN são iguais a [70,70] e [18,18], respectivamente. Os lugares *in_A, W_A, B_A* e *out_A* representam, respectivamente, a entrada no estado A, o estado de pior caso, o estado de melhor caso e a saída do estado A. Caso as restrições de tempo forem omitidas dos estado simples, então esses estados são mapeados em transições-PN cujos valores relacionados ao tempo máximo e mínimo são iguais a zero (ver Figura 4.3) [AND09].

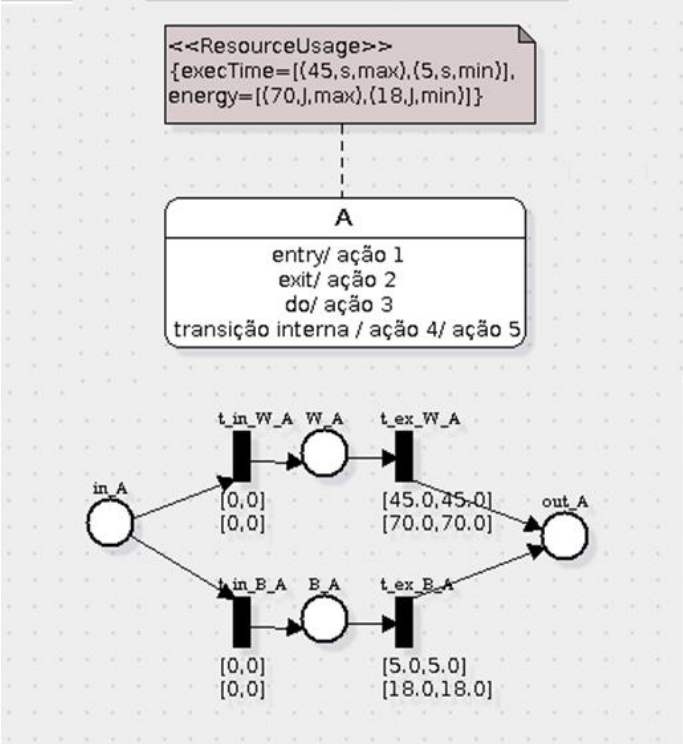

Figura 4.2: Mapeamento do estado simples com restrições de tempo e anotações de energia

As transições *t_in_W_A* e *t_in_B_A* são adicionadas ao modelo ETPN, para que transições-PN com retardos maiores sejam capazes de disparar, devido à semântica de disparo adotada das Redes de Petri Temporizadas (semântica de disparo *strong* (ver Seção 2.4.1)). Por essa razão, um intervalo estreito igual [0,0] é atribuído a essas transições-PN. Além disso, as transições-PN *t_ex_W_A* e *t_ex_B_A* do modelo ETPN (ver Figura 4.2) são atribuídas com um intervalo estreito de, respectivamente, [45,45] e [5,5], porque o espaço de estados é substancialmente menor do aquele gerado, se apenas uma única transição-PN fosse adotada com o intervalo igual a [5,45].

Definição 4.1. (Estados com restrições) O modelo do estado com restrições é uma ETPN definida como $ER = (P^{ER}, T^{ER}, F^{ER}, W^{ER}, m_0^{ER}, I^{ER}, \varepsilon^{ER})$, onde:

- $P^{ER} = \{in\_A, W\_A, B\_A, out\_A\};$
- $T^{ER} = \{t\_in\_W\_A, t\_in\_B\_A, t\_ex\_W\_A, t\_ex\_B\_A\};$
- $F^{ER} = \{(in\_A, t\_in\_W\_A), (in\_A, t\_in\_B\_A), (t\_in\_W\_A, W\_A), (t\_in\_B\_A, B\_A), (W\_A, t\_ex\_W\_A), (B\_A, t\_ex\_B\_A), (t\_ex\_W\_A, out\_A), (t\_ex\_B\_A, out\_A)\};$
- $W^{ER}(f) = 1, \forall f \in F^{ER};$
- $m_0^{ER}(p) = 0, \forall p \in P^{ER};$
- $I^{ER}(t) = (EFT^{ER}(t), LFT^{ER}(t)), \forall t \in T^{ER}$, onde:

$$EFT^{ER}(t) = LFT^{ER}(t) = \begin{cases} X_E^{ER}, \text{ se } t = t\_ex\_W\_A; \\ X_L^{ER}, \text{ se } t = t\_ex\_B\_A; \\ 0, \text{ caso contrário;} \end{cases}$$

- $\varepsilon^{ER}(t) = \begin{cases} Y_E^{ER}, \text{ se } t = t\_ex\_W\_A; \\ Y_L^{ER}, \text{ se } t = t\_ex\_W\_A; \\ 0, \text{ caso contrário.} \end{cases}$

onde X_E^{ER} e X_L^{ER} são os tempos de execução no melhor e pior caso associados às transições *t_ex_B_A* e *t_ex_W_A*, respectivamente, e Y_E^{ER} e Y_L^{ER} são os consumos de energia no melhor e pior caso associados às transições *t_ex_B_A* e t_ex_W_A, respectivamente [AND09].

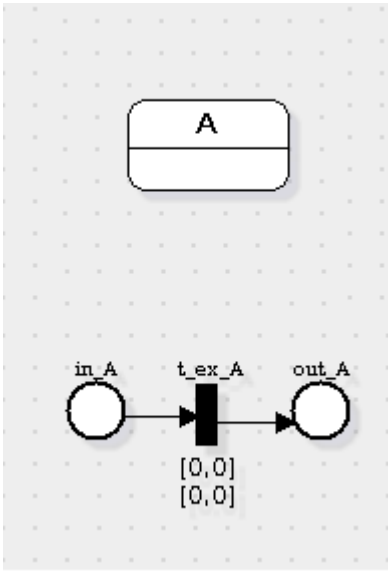

Figura 4.3: Mapeamento do estado simples

Definição 4.2. (Estados sem restrições) O modelo do estado sem restrições é uma ETPN definida como $ESR = (P^{ESR}, T^{ESR}, F^{ESR}, W^{ESR}, m_0^{ESR}, I^{ESR}, \varepsilon^{ESR})$, onde:

- $P^{ESR} = \{in\_A, out\_A\};$
- $T^{ESR} = \{t\_ex\_A\};$
- $F^{ESR} = \{(in\_A, t\_ex\_A), (t\_ex\_A, out\_A)\};$
- $W^{ESR}(f) = 1, \forall f \in F^{ESR};$
- $m_0^{ESR}(p) = 0, \forall p \in P^{ESR};$
- $I^{ESR}(t) = (EFT^{ESR}(t), LFT^{ESR}(t)), \forall t \in T^{ESR}$, *onde:*

$EFT^{ESR}(t) = LFT^{ESR}(t) = 0 \ \forall t \in TE^{SR}$

- $\varepsilon^{ESR}(t) = 0 \ \forall t \in T^{ESR}$

4.2 TRANSIÇÕES INTERNAS

As transições internas não causam mudança de estado. Geralmente, o evento associado à uma transição interna é acompanhado de uma ação cuja execução não provoca nenhuma mudança no estado da entidade.

A Figura 4.4 mostra um exemplo de uma transição interna causada pelo disparo do evento *dar troco*. Quando o estado *Máquina de Refrigerante* é assumido, as ações *Colocar Dinheiro* e *Colocar Copo* são executadas. Em seguida, a ação *Colocar Refrigerante no Copo* é executada, enquanto a entidade se encontra nesse estado. Durante a execução dessa última ação, é possível executar também a transição interna *dar troco,* a qual irá disparar a ação *Devolver Troco*. Durante a execução da ação *Devolver Troco*, nenhuma outra ação poderá ser executada, além dessa própria ação e de *Colocar Refrigerante no Copo.* Por último, a ação *Retirar Copo* é executada, antes da entidade mudar de estado.

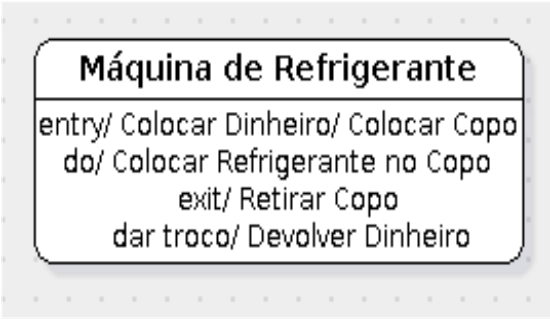

Figura 4.4: Exemplo de uma transição interna

4.3 MAPEAMENTO DAS TRANSIÇÕES

Ao contrário do que foi visto anteriormente, uma transição representa o relacionamento entre dois estados, onde o disparo da mesma indica que a entidade que está no primeiro estado, irá passar para o segundo estado. Uma transição é, graficamente, representada por uma seta que aponta para o seu destino [GUE04].

A Figura 4.5 apresenta um exemplo de um diagrama de estados que contém uma transição. Os estados representam o processo de consulta pelo CPF e atualização do cadastro. Enquanto a entidade está no estado *Consultar Pessoa*, o método *ConcCPF()* é executado. Após isso, ocorre a transição de estados, na qual a entidade assume o novo estado *Atualizando Pessoa*. Nesse estado, a entidade executa o método *Gravar()*. Note que durante a transição de estados, as anotações de MARTE são usadas para representar as restrições do sistema sendo modelado. Em outras palavras, a transição de estados possui o tempo de execução no pior caso de 10 segundos e no melhor caso de 3 segundos. Além disso, essa transição possui um consumo de energia no pior caso de 50 *joules*, e no melhor caso de 30 *joules*.

A Figura 4.6 ilustra o mapeamento de uma transição do diagrama de estados (transição-DE) com restrições de tempo e anotações de energia. A transição-DE, que sai do estado *A* com destino o estado *B*, é mapeada em duas transições-PN *(t_ex_W_t1* e *t_ex_B_t1)*, as quais possuem intervalos estreitos, que representam o tempo de execução no pior e melhor caso, iguais a, respectivamente, [25,25] e [15,15]. O mapeamento dessa transição gera um modelo ETPN exatamente igual ao modelo gerado pelo mapeamento do estado simples com restrições de tempo e anotações de energia. Já transições-DE,

que não possuem restrições de tempo, são mapeadas em transições-PN cujos tempos máximo e mínimo possuem valor iguais a zero (ver Figura 4.7) [AND09].

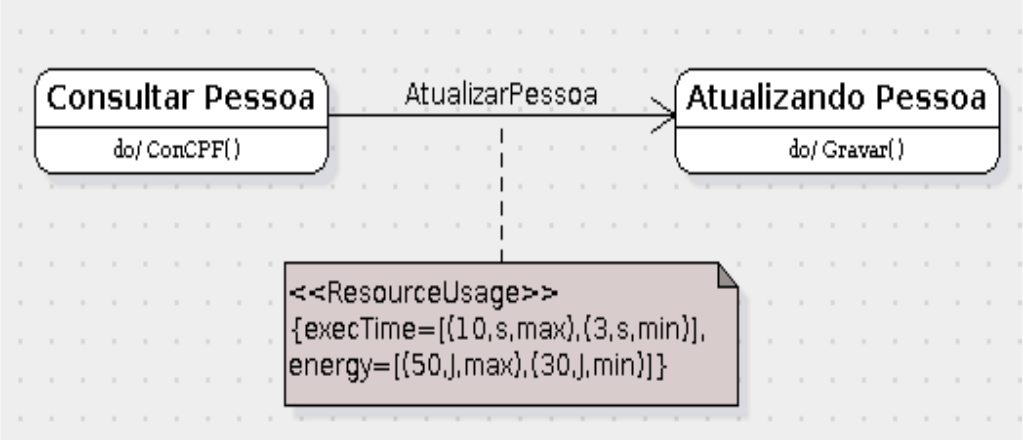

Figura 4.5: Exemplo de uma transição

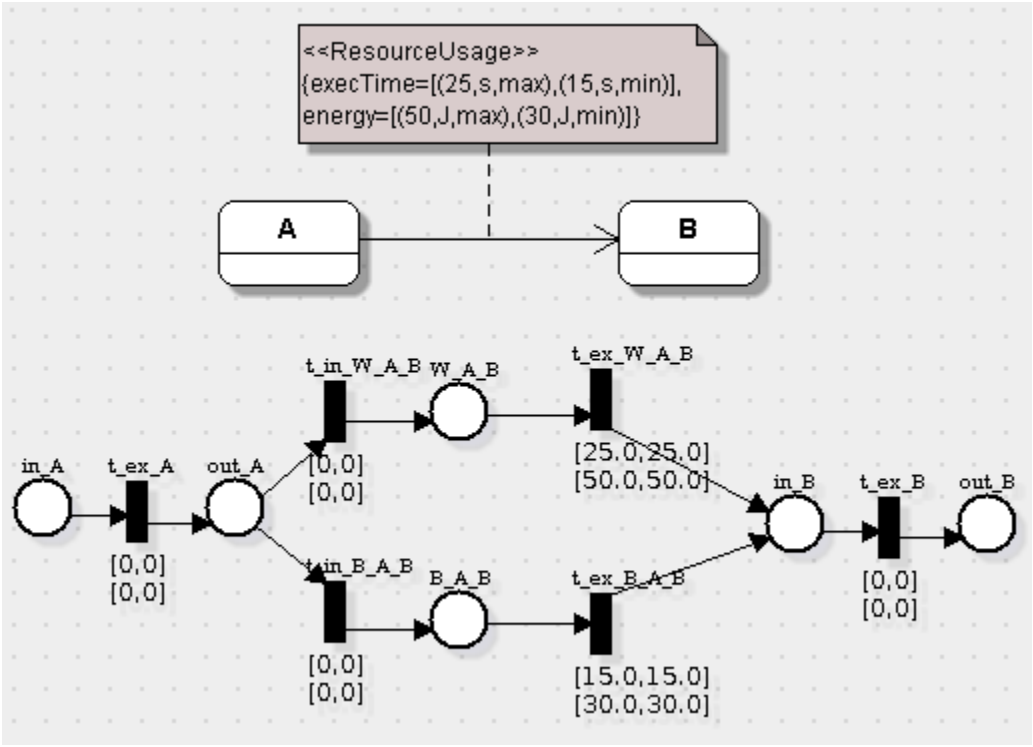

Figura 4.6: Mapeamento das transições com restrições de tempo e anotação de energia

Definição 4.3. (Transições com restrições) O modelo da transição com restrições é uma ETPN definida como $TR = (P^{TR}, T^{TR}, F^{TR}, W^{TR}, m_0^{TR}, I^{TR}, \varepsilon^{TR})$, onde:

- $P^{TR} = \{out\_A, W\_t1, B\_t1, in\_B\};$
- $T^{TR} = \{t\_in\_W\_t1 , t\_in\_B\_t1, t\_ex\_W\_t1, t\_ex\_B\_t1 \};$
- $F^{TR} = \{(out\_A, t\_in\_W\_t1), (out\_A, t\_in\_B\_t1), (t\_in\_W\_t1, W\_t1), (t\_in\_B\_t1, B\_t1), (W\_t1, t\_ex\_W\_t1), (B\_t1, t\_ex\_B\_t1), (t\_ex\_W\_t1, in\_B), (t\_ex\_B\_t1, in\_B)\};$
- $W^{TR}(f) = 1, \forall f \in F^{TR};$
- $m_0^{TR}(p) = 0, \forall p \in P^{TR};$
- $I^{TR}(t) = (EFT^{TR}(t), LFT^{TR}(t)), \forall t \in T^{TR}$, *onde:*

$$EFT^{TR}(t) = LFT^{TR}(t) = \begin{cases} X_E^{TR}, \text{ se } t = t\_ex\_W\_t1; \\ X_L^{TR}, \text{ se } t = t\_ex\_B\_t1; \\ 0, \text{ caso contrário;} \end{cases}$$

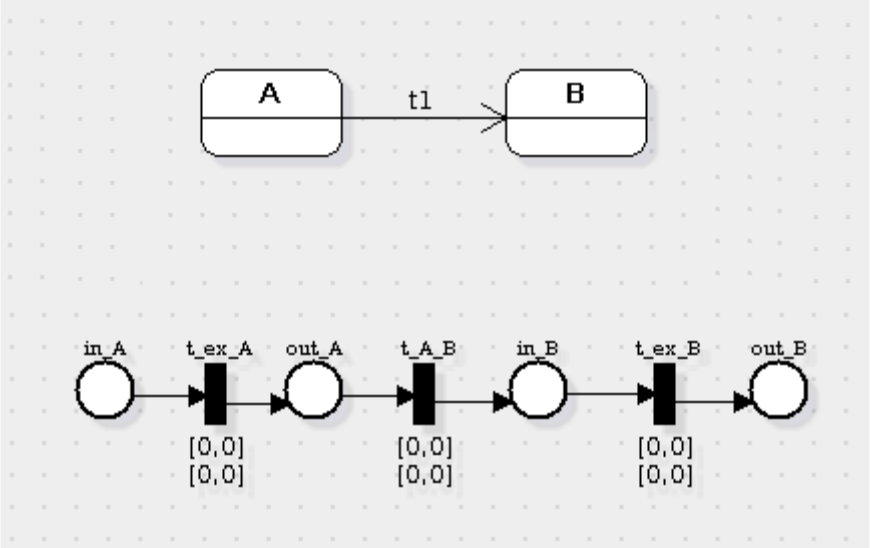

Figura 4.7: Mapeamento da transição

- $\varepsilon^{TR}(t) = \begin{cases} Y_E^{TR}, \text{ se } t = t\_ex\_W\_t1; \\ Y_L^{TR}, \text{ se } t = t\_ex\_W\_t1; \\ 0, \text{ caso contrário.} \end{cases}$

onde X_E^{TR} e X_L^{TR} são os tempos de execução no melhor e pior caso associados às transições *t_ex_B_t1* e *t_ex_W_t1*, respectivamente, e Y_E^{TR} e Y_L^{TR} são os consumos de energia no melhor e pior caso associados às transições *t_ex_B_t1* e t_ex_W_t1, respectivamente [AND09].

Definição 4.4. (Transição sem restrições) O modelo da transição sem restrições é uma ETPN definida como $TSR = (P^{TSR}, T^{TSR}, F^{TSR}, W^{TSR}, m_0^{TSR}, I^{TSR}, \varepsilon^{TSR})$, onde:

- $P^{TSR} = \{out\_A, in\_B\}$;
- $T^{TSR} = \{t\_ex\_t1\}$;
- $F^{TSR} = \{(out\_A, t\_ex\_t1), (t\_ex\_t1, in\_B)\}$;
- $W^{TSR}(f) = 1, \forall f \in F^{TSR}$;
- $m_0^{TSR}(p) = 0, \forall p \in P^{TSR}$;
- $I^{TSR}(t) = (EFT^{TSR}(t), LFT^{TSR}(t)), \forall t \in T^{TSR}$, onde:

$EFT^{TSR}(t) = LFT^{ESR}(t) = 0 \ \forall t \in T^{TSR}$

- $\varepsilon^{TSR}(t) = 0 \ \forall t \in T^{TSR}$

4.4 MAPEAMENTO DAS AUTO-TRANSIÇÕES

Uma auto-transição é uma transição que sai do estado atual e retorna ao mesmo estado, ocasionando uma reentrada no mesmo estado. Diferentemente das transições internas, as auto-transições saem do estado podendo executar ou não alguma ação, e depois retornam ao estado de onde saíram. Note que, uma vez retornada ao estado de origem, a auto-transição não será mais executada.

O mapeamento de uma auto-transição é ilustrado na Figura 4.8. Nesse caso, existe uma auto-transição intitulada *sel1*, cujo tempo de execução no pior e melhor caso são, respectivamente, iguais a 30µs e 10µs. As regras de mapeamento para esse tipo de transição são semelhantes às regras apresentadas anteriormente para o mapeamento de transições de estados, porém o modelo ETPN da auto-transição retorna ao estado A através da transição-PN *t_sel1_A* [AND09].

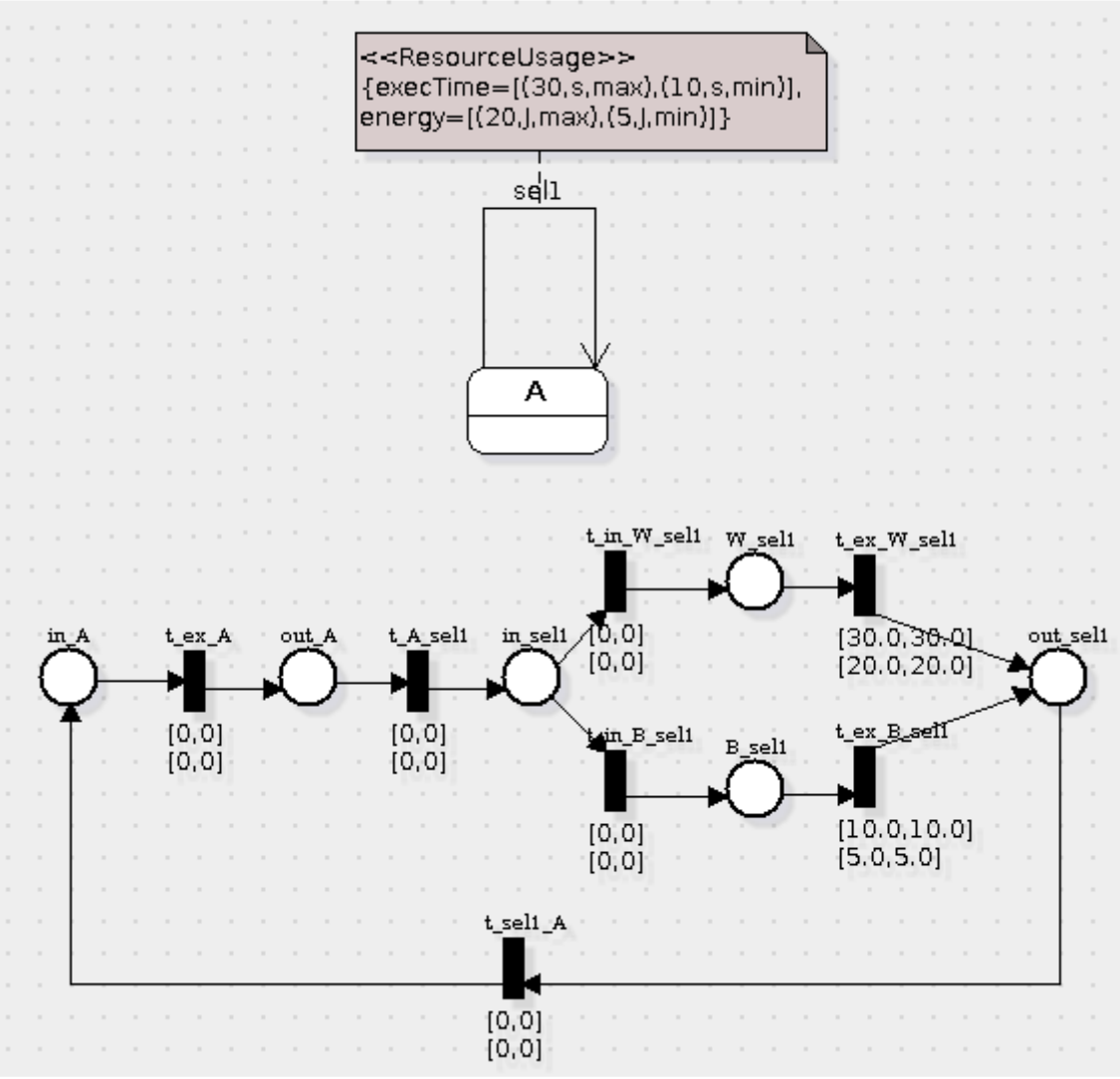

Figura 4.8: Mapeamento das auto-transições

Definição 4.5. (Auto-transições com restrições) O modelo da auto-transição com restrições é uma ETPN definida como $ATR = (P^{ATR}, T^{ATR}, F^{ATR}, W^{ATR}, m_0^{ATR}, I^{ATR}, \varepsilon^{ATR})$, onde:

- $P^{ATR} = \{in\_A, out\_A, in\_sel1, W\_sel1, B\_sel1, out\_sel1\};$

- T^{ATR} = {t_ex_A, t_A_sel1, t_in_W_sel1, t_in_B_sel1, t_ex_W_sel1, t_ex_B_sel1, t_sel1_A};
- F^{ATR} = {(in_A, t_ex_A), (t_ex_A, out_A), (out_A, t_A_sel1), (t_A_sel1, in_sel1), (in_sel1, t_in_W_sel1), (in_sel1, t_in_B_sel1), (t_in_W_sel1, W_sel1), (t_in_B_sel1, B_sel1), (W_sel1, t_ex_W_sel1), (B_sel1, t_ex_B_sel1), (t_ex_W_sel1, out_sel1), (t_ex_B_sel1, out_sel1), (out_sel1, t_sel1_A), (t_sel1_A, in_A)};
- $W^{ATR}(f) = 1, \forall f \in F^{ATR}$;
- $m_0^{ATR}(p) = 0, \forall p \in P^{ATR}$;
- $I^{ATR}(t) = (EFT^{ATR}(t), LFT^{ATR}(t)), \forall t \in T^{ATR}$, onde:

$$EFT^{ATR}(t) = LFT^{ATR}(t) \begin{cases} X_E^{ATR}, \text{ se } t = t\_ex\_W\_sel1; \\ X_L^{ATR}, \text{ se } t = t\_ex\_B\_sel1; \\ 0, \text{ caso contrário;} \end{cases}$$

- $\varepsilon^{TSR}(t) = 0 \ \forall t \in T^{ATR}$

onde X_E^{ATR} *e* X_L^{ATR} são os tempos de execução no melhor e pior caso associados às transições *t_ex_B_sel1* e *t_ex_W_sel1*, respectivamente [AND09].

Definição 4.6. (Auto-transições sem restrições) O modelo do auto-estado sem restrições é uma ETPN definida como $ASR = (P^{ASR}, T^{ASR}, F^{ASR}, W^{ASR}, m_0^{ASR}, I^{ASR}, \varepsilon^{ASR})$, onde:

- P^{ASR} = {in_A, out_A };
- T^{ASR} = {t_ex_A, t_sel1_A};
- F^{ASR} = {(in_A, t_ex_A), (t_ex_A, out_A), (out_A, t_sel1_A), (t_sel1_A, in_A)};
- $W^{ASR}(f) = 1, \forall f \in F^{ASR}$;
- $m_0^{ASR}(p) = 0, \forall p \in P^{ASR}$;
- $I^{ASR}(t) = (EFT^{ASR}(t), LFT^{ASR}(t)), \forall t \in T^{ASR}$, onde:
  $EFT^{ASR}(t) = LFT^{ASR}(t) = 0 \ \forall t \in TASR$
- $\varepsilon^{ASR}(t) = 0 \ \forall t \in T^{ASR}$

4.5 MAPEAMENTO DOS ESTADOS COMPOSTOS

O diagrama de estados da SysML admite o conceito de estados aninhados, ou seja, estados que contém outros estados. Eles são chamados de estados compostos, e sua inserção no modelo permite visões de alto nível do comportamento do estado da entidade. Estados compostos podem ser decompostos em: estados sequenciais e estados concorrentes.

Estados compostos sequenciais são aqueles que podem ser divididos em estados simples. A Figura 4.9 apresenta um exemplo de um estado composto sequencial. O estado *Alterando Pessoa* foi sub-dividido em dois estados *(Validando CPF e Gravando Pessoa),* onde ambos são utilizados para detalhar o estado *Alterando Pessoa*. Na Figura 4.10, é apresentado um exemplo do mapeamento de um estado composto sequencial no modelo ETPN. Como pode ser observado, o estado *B* foi sub-dividido em dois subestados *(C e D)*, sendo que o sub-estado *C* possui restrições de tempo. As regras de mapeamento dos subestados são semelhantes às regras mencionadas anteriormente para os estados simples e transições-DE [AND09].

Definição 4.7. (Estados compostos sequenciais com restrições) O modelo dos estados compostos sequenciais com restrições é uma ETPN definida como $CON = (P^{CON}, T^{CON}, F^{CON}, W^{CON}, m_0^{CON}, I^{CON}, \varepsilon^{CON})$, onde:

- $P^{CON} = \{out\_A, in\_C, W\_C, B\_C, out\_C, in\_D, out\_D, in\_E, out\_E\};$
- $T^{CON} = \{t\_ex\_t1, t\_in\_W\_C, t\_in\_B\_C, t\_ex\_W\_C, t\_ex\_B\_C, t\_ex\_t2, t\_ex\_D, t\_ex\_t3, t\_ex\_E\};$
- $F^{CON} = \{(out\_A, t\_ex\_t1), (t\_ex\_t1, in\_C), (in\_C, t\_in\_W\_C), (in\_C, t\_in\_B\_C), (t\_in\_W\_C, W\_C), (t\_in\_B\_C, B\_C), (W\_C, t\_ex\_W\_C), (B\_C, t\_ex\_B\_C), (t\_ex\_W\_C, out\_C), (t\_ex\_B\_C, out\_C), (out\_C, t\_ex\_t2), (t\_ex\_t2, in\_D), (in\_D, t\_ex\_D), (t\_ex\_D, out\_D), (out\_D, t\_ex\_t3), (t\_ex\_t3, in\_E), (in\_E, t\_ex\_t3), (t\_ex\_E, out\_E)\};$
- $W^{CON}(f) = 1, \forall f \in F^{CON};$
- $m_0^{CON}(p) = 0, \forall p \in P^{CON};$
- $I^{CON}(t) = (EFT^{CON}(t), LFT^{CON}(t)), \forall t \in T^{CON}$, *onde:*

Figura 4.9: Exemplo de um estado composto sequencial

$$EFT^{CON}(t) = LFT^{CON}(t) \begin{cases} X_E^{CON}, \text{ se } t = t\_ex\_W\_C; \\ X_L^{CON}, \text{ se } t = t\_ex\_B\_C; \\ 0, \text{ caso contrário;} \end{cases}$$

- $\varepsilon^{TSR}(t) = 0 \ \forall t \in T^{CON}$

onde X_E^{CON} *e* X_L^{CON} são os tempos de execução no melhor e pior caso associados às transições *t_ex_B_C* e *t_ex_W_C*, respectivamente [AND09].

O estado composto concorrente é um estado que divide seus estados em regiões, onde cada região contém diagramas distintos, que podem ser executados concorrentemente. A Figura 4.11 ilustra um exemplo de um diagrama de estados que contém um estado composto concorrente. Nesse exemplo, para que o carro arranque é

necessário que as ações dos estados *Soltando embreagem* e *Pressionando Acelerador (Soltar Embreagem e Pressionar Acelerador)* ocorram em paralelo.

Um exemplo do mapeamento dos estados concorrentes é apresentado na Figura 4.12. Esse tipo de mapeamento é adotado para modelar atividades concorrentes, onde duas transições-PN *t_in_par_C_D* e *t_sin_C_D* são usadas para representar, respectivamente, o início e o fim das atividades concorrentes. Além disso, o diagrama de cada região é mapeado de acordo com as regras de mapeamento mencionadas anteriormente [AND09].

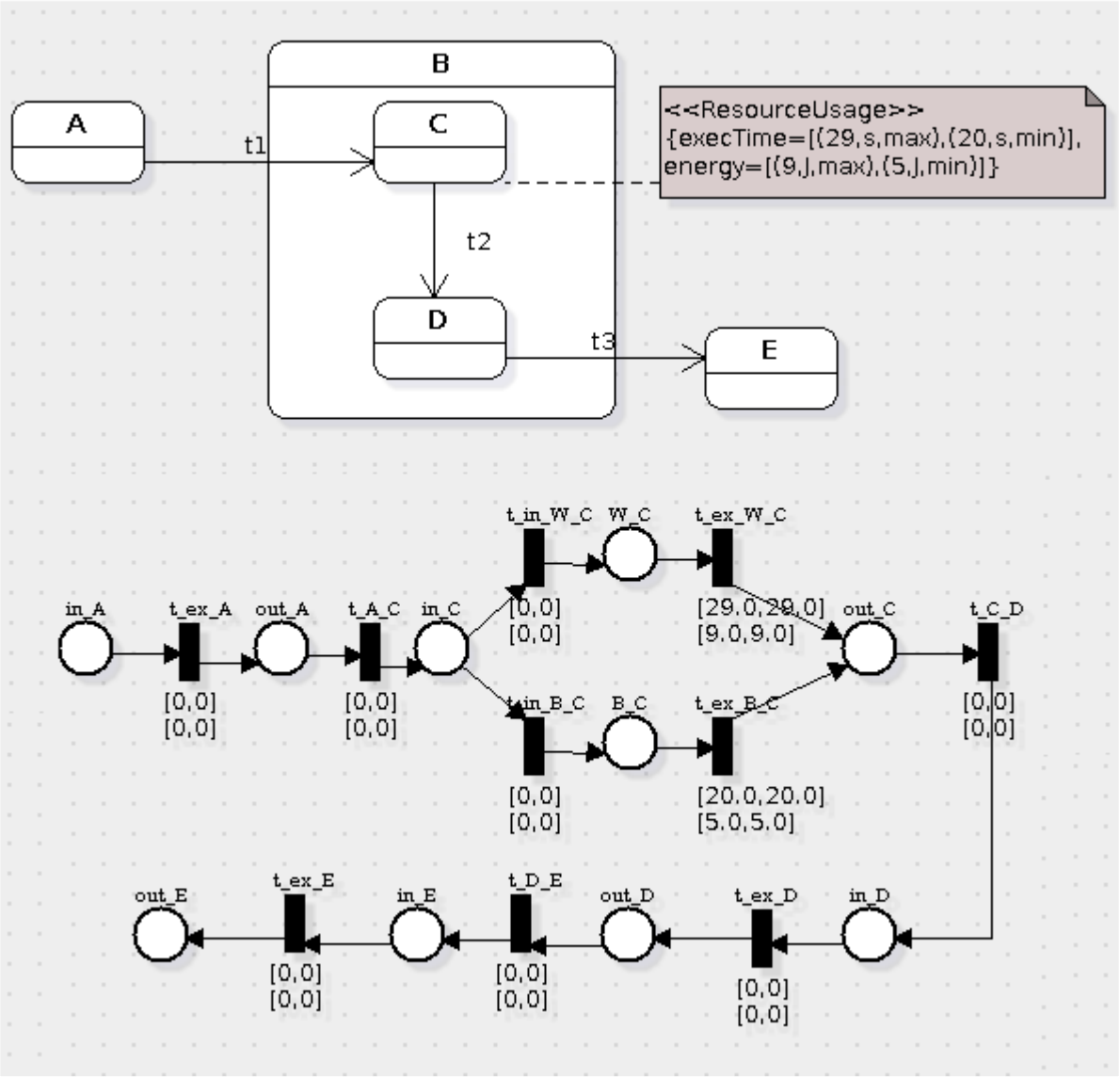

Figura 4.10: Mapeamento de um estado composto sequencial

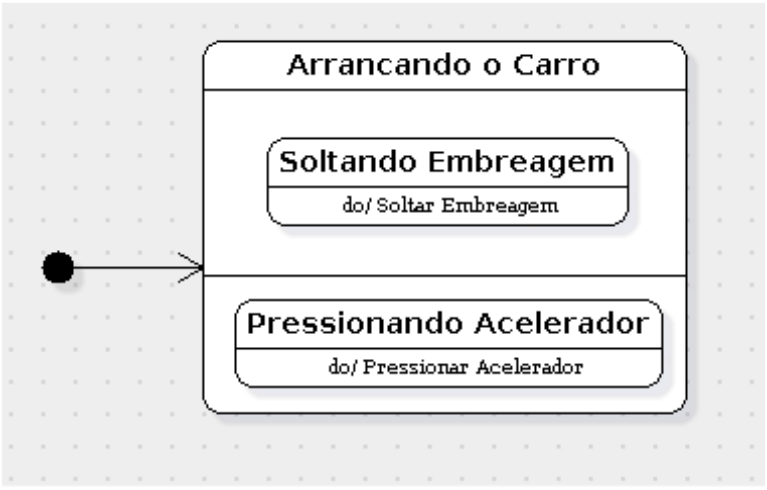

Figura 4.11: Exemplos de estados concorrentes

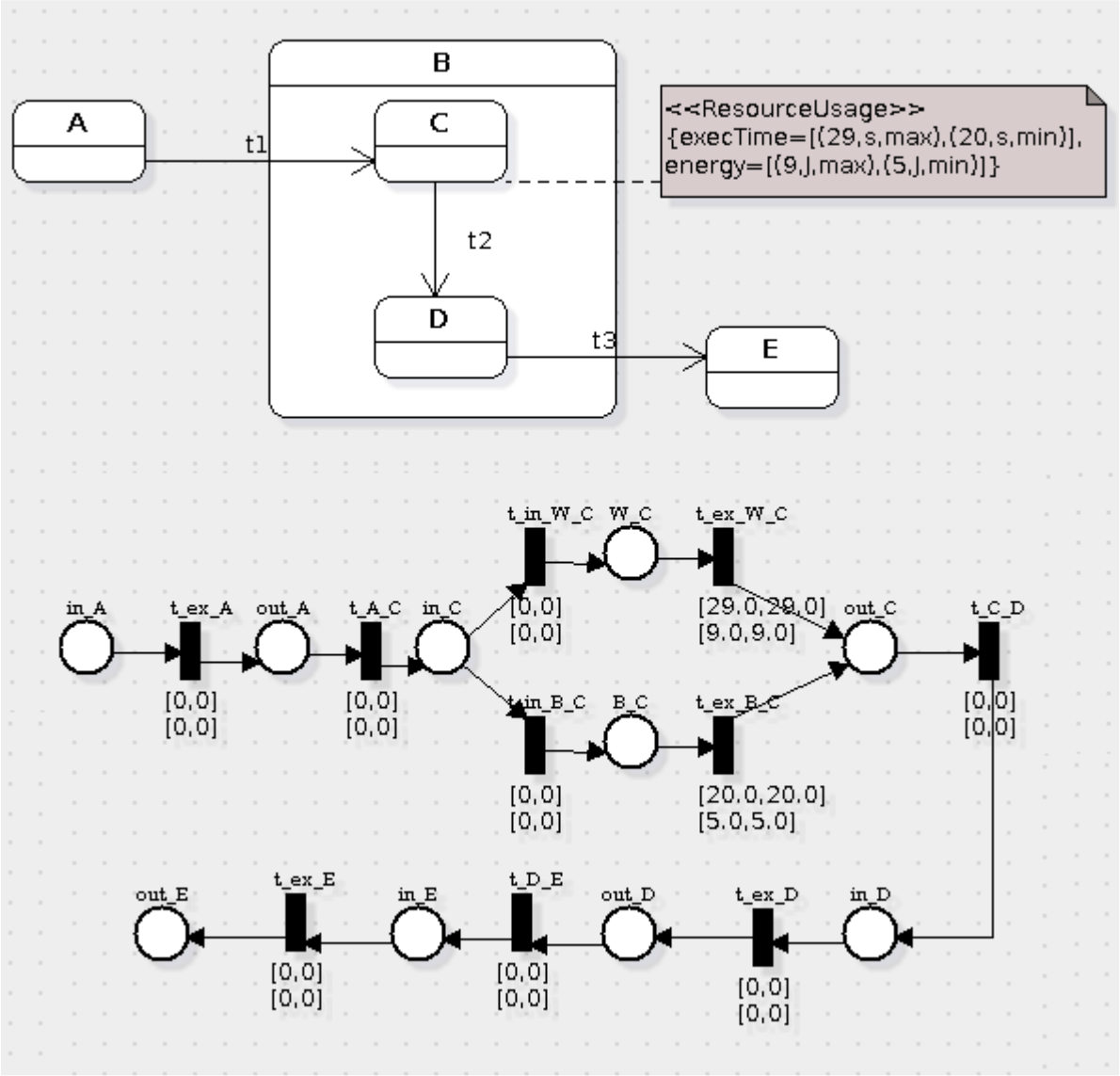

Figura 4.12: Mapeamento dos estados concorrentes

Definição 4.7. (Estados compostos concorrentes com restrições) O modelo dos estados compostos concorrentes com restrições é uma ETPN definida como $CON = (P^{CON}, T^{CON}, F^{CON}, W^{CON}, m_0^{CON}, I^{CON}, \varepsilon^{CON})$, onde:

- $P^{CON} = \{out\_A, in\_C, W\_C, B\_C, out\_C, in\_D, out\_D, in\_E, out\_E\};$
- $T^{CON} = \{t\_ex\_t1, t\_in\_W\_C, t\_in\_B\_C, t\_ex\_W\_C, t\_ex\_B\_C, t\_ex\_t2, t\_ex\_D, t\_ex\_t3, t\_ex\_E\};$
- $F^{CON} = \{(out\_A, t\_ex\_t1), (t\_ex\_t1, in\_C), (in\_C, t\_in\_W\_C), (in\_C, t\_in\_B\_C), (t\_in\_W\_C, W\_C), (t\_in\_B\_C, B\_C), (W\_C, t\_ex\_W\_C), (B\_C, t\_ex\_B\_C), (t\_ex\_W\_C, out\_C), (t\_ex\_B\_C, out\_C), (out\_C, t\_ex\_t2), (t\_ex\_t2, in\_D), (in\_D, t\_ex\_D), (t\_ex\_D, out\_D), (out\_D, t\_ex\_t3), (t\_ex\_t3, in\_E), (in\_E, t\_ex\_t3), (t\_ex\_E, out\_E)\};$
- $W^{CON}(f) = 1, \forall f \in F^{CON};$
- $m_0^{CON}(p) = 0, \forall p \in P^{CON};$
- $I^{CON}(t) = (EFT^{CON}(t), LFT^{CON}(t)), \forall t \in T^{CON}$, onde:
- $$EFT^{CON}(t) = LFT^{CON}(t) \begin{cases} X_E^{CON}, \text{ se } t = t\_ex\_W\_C; \\ X_L^{CON}, \text{ se } t = t\_ex\_B\_C; \\ 0, \text{ caso contrário;} \end{cases}$$
- $\varepsilon^{TSR}(t) = 0 \ \forall t \in T^{CON}$

onde X_L^{CON} e X_E^{CON} são os tempos de execução no melhor e pior caso associados às transições *t_ex_B_C* e *t_ex_W_C*, respectivamente [AND09].

4.6 MAPEAMENTO DOS ESTADOS INICIAL E FINAL

O estado inicial é um pseudo-estado cuja função é indicar o ponto de partida no qual o diagrama de estados ou os estados compostos serão analisados. Graficamente, o estado inicial é representado por um círculo preenchido (ver Figura 4.13). Esse estado é mapeado em um lugar (*staIni_A*) com marcação inicial igual a um token. Os tokens são usados nos modelos para simular o comportamento dinâmico dos sistemas. Além disso, a transição-PN *t_in_A* é usada para representar a transição-DE entre o estado incial e o estado simples *A* [AND09] (ver Figura 4.13).

O estado final é um pseudo-estado cuja função é determinar tanto o fim do diagrama de estados quanto dos estados compostos. Graficamente, o estado final é representado por um círculo não preenchido envolvendo outro círculo preenchido (ver Figura 4.14). Esse estado é mapeado em um lugar *(endSta_A)* no modelo ETPN. A presença de um token no lugar *endSta_A,* representa o fim do diagrama de estados ou do estado composto. A transição-PN *t_end_A* é usada para representar a transição-DE entre o estados simples *A* e o estado final, como é demonstrado na Figura 4.14 [AND09].

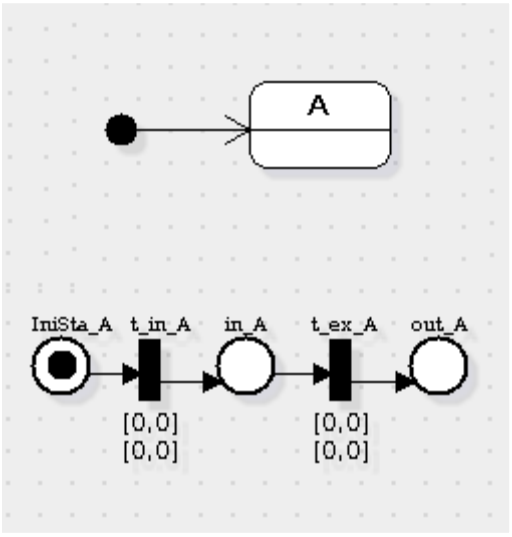

Figura 4.13: Mapeamento do estado inicial

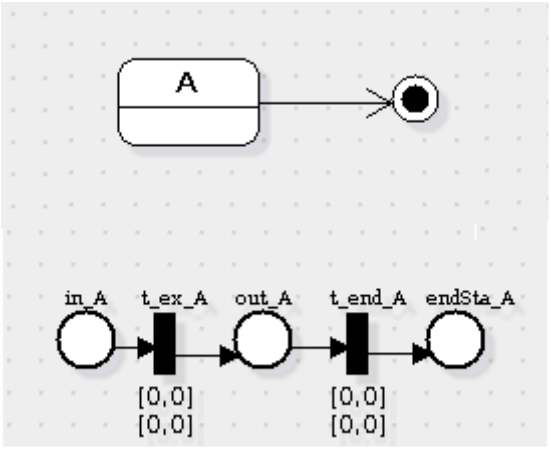

Figura 4.14: Mapeamento do estado final

4.7 CONSIDERAÇÕES FINAIS

Nesse capítulo, foram vistos como os componentes do Diagrama de Estados da SysML são mapeados em Redes de Petri. Com isso, em posse de um Diagrama de Estados, é possível obter um modelo formal equivalente e aplicar nele técnicas matemáticas sólidas para avaliação de desempenho.

CAPÍTULO 5

ESTUDO DE CASO

Neste capítulo, são apresentados dois estudos de caso para demonstrar a aplicabilidade da ferramenta Calau. No primeiro, será realizado um estudo sobre o controle de rotação usado para limitar a velocidade de automóveis. Depois, será feito um estudo sobre uma impressora térmica. Para esses exemplos serão realizados análises e verificações dos modelos ETPNs gerados pelo processo de mapeamento.

5.1 CONTROLE DE ROTAÇÃO DE MOTOR DE UM AUTOMÓVEL

O sistema de controle de rotação de motor é usado para limitar a velocidade de um veículo. Para isso, os sistemas de freios são ativados, caso o automóvel atinja uma velocidade específica. Mais detalhes podem ser encontrados em [AMN+10]. A Figura 5.1, apresenta o diagrama de estados do controle de rotação. Como pode ser observado, o sistema é iniciado no estado *Normal*. Nesse estado, a velocidade do carro é verificada constantemente. Se o carro ultrapassar o limite máximo de velocidade, o sistema assumirá o estado *Alarm On*, onde indica que o alarme e os freios serão ativados. Em seguida, a velocidade é verificada mais uma vez. Se a velocidade estiver dentro do limite permitido, o sistema assumirá o estado *Alarm Off*, no qual, o alarme e os freios serão desligados. Após isso, o sistema reassume o estado normal.

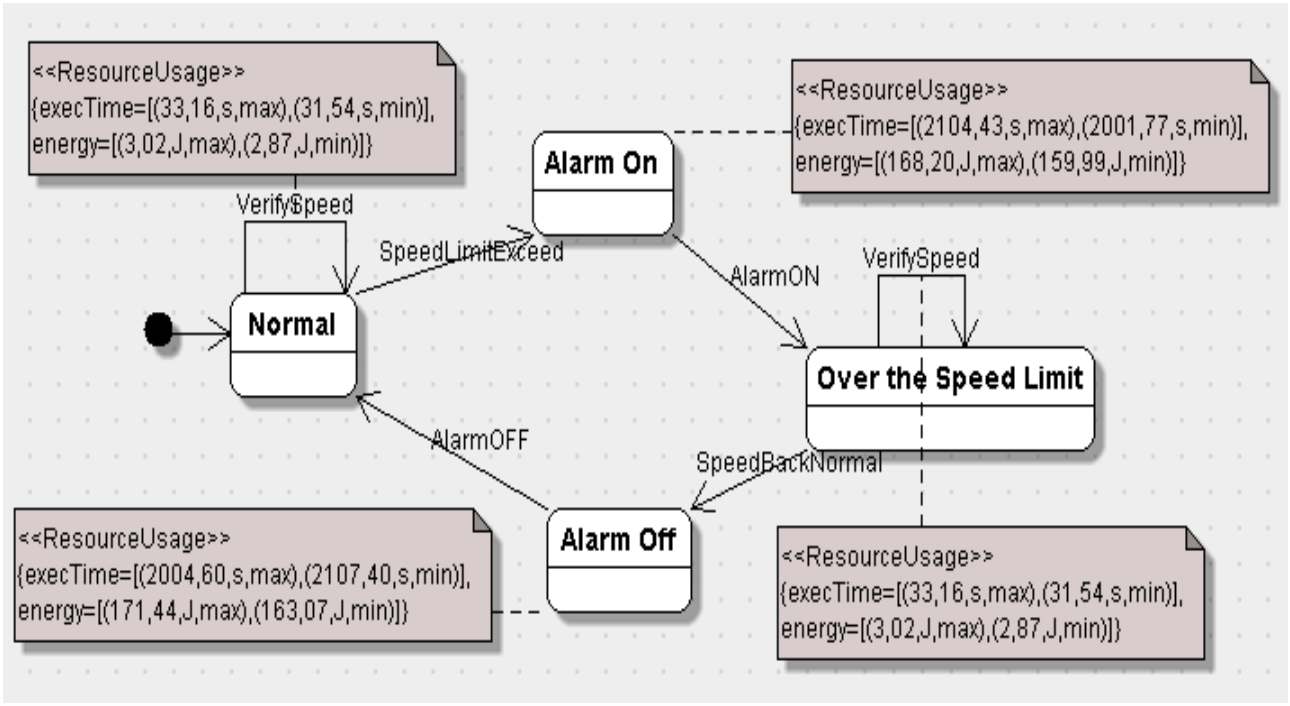

Figura 5.1: Diagrama de Estados do Controle de Rotação

A Figura 5.2 apresenta o modelo ETPN do controle de rotação. Esse modelo foi obtido utilizando o mapeamento automático fornecido por Calau. Foram computadas as estimativas quanto ao tempo de execução e consumo de energia. Os resultados estão detalhados na Figura 5.3. O *BCET* calculado para o componente de controle de rotação foi 4069.45 *ms* e o *WCET* foi 4278.14 *ms*. Além disso, seu consumo de energia também

foi calculado. Os resultados obtidos para o melhor e pior caso foram 328,80 *μJ* e 345,68 *μJ*, respectivamente. Também foi realizada a análise qualitativa do modelo ETPN. Para isso são gerados arquivos de entrada do INA a partir do modelo. Como já mencionado, é importante explicar as propriedades qualitativas dentro do contexto do problema, porque elas podem revelar características importantes do sistema sendo modelado. A análise mostra que o sistema modelado possui as seguintes propriedades [Mur89]:

- ***Deadlock Freedom*** - A análise do controle de rotação indica que o sistema é livre de *deadlock*. O que significa que o sistema não tem qualquer estado morto.
- ***Liveness*** - Uma rede de Petri é dita ser livre, se a partir de uma marcação inicial, é possível, disparar qualquer transição da rede através de uma sequência de disparos, não importa qual marcação seja alcançada. Isto significa que uma rede de Petri livre garante a ausência de *deadlock*, não importando qual a sequência de disparo seja escolhido. O modelo ETPN do controle de rotação não é livre devido à um conjunto de transições de rede de Petri relacionadas às atividades que só são executadas durante a fase inicial. Se essas atividades forem desconsideradas, o componente pode ser considerado *live*.
- **Reversível** – A rede não é reversível porque ela não pode retornar ao seu estado inicial depois da partida, de modo que o estado inicial só pode ser alcançado se o sistema é desligado e reiniciado. No entanto, há algumas marcações (estados) que podem ser alcançado novamente disparando uma sequência de PN-transições (eventos).

5.2 IMPRESSORA TÉRMICA

A impressora térmica produz uma imagem impressa no papel, no momento em que ele está sendo aquecido pela cabeça de impressão térmica, que é controlada por um microprocessador. Essas impressoras são bastante usadas por serem econômicas, silenciosas, compactas e não precisarem de reposição de tinta. São empregadas principalmente para imprimir recibos de caixas eletrônicos, etiquetas, códigos barras, entre outras finalidades. A Figura 5.4 apresenta dois exemplos de impressoras térmicas.

O funcionamento da impressora térmica se dá por monitorar botões e sensores, controlar os *leds,* a cabeça de impressão e o motor de passo, dentre outras funções. A Figura 5.5 apresenta um esquema simplificado dos processos de uma impressora e suas interações. Para fins deste trabalho, apenas a tarefa do controlador de impressão será adotada. Essa tarefa é a principal dentre as demais, já que é encarregada de disparar o processo que controla o motor de avanço do papel, e no momento certo, liberar o processo que controla o acionamento da cabeça térmica, caso existam linhas de impressão [AMF+09].

Na Figura 5.6, é apresentado o diagrama de estados do controlador de impressão com restrições de tempo e anotações de energia. Essas informações são relacionadas à plataforma de *hardware Philips LPC2106*. Conforme descrito no modelo, caso existam dados para serem impressos, o controlador de impressão os envia para o controlador da *SPI (Serial Peripheral Interface),* e após isso, fica aguardando uma sinalização de retorno quando todos os dados forem recebidos. Finalizada essa etapa, é acionado o controlador do motor. Assim que o motor avançar a quantidade de passos pré-estabelecidos, o seu controlador informará ao controlador de impressão esse fato. Nesse momento, a cabeça térmica está pronta para imprimir os dados no papel. Para isso, um sinal *strobe*, que sinaliza quando as informações podem ser carregadas na cabeça de impressão, é enviado ao controlador da cabeça, o qual informará ao controlador de impressão quando

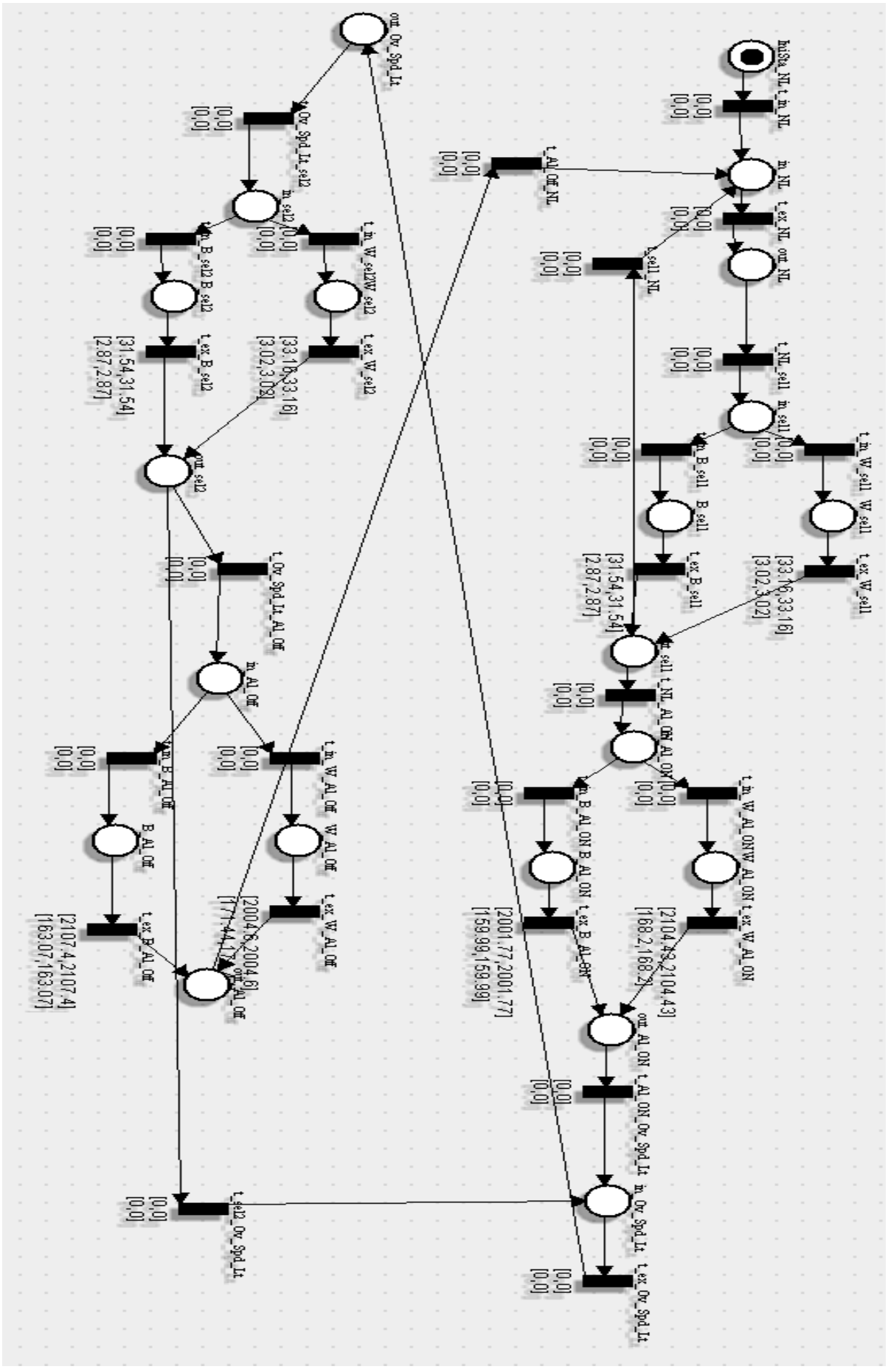

Figura 5.2: Modelo ETPN do controle de rotação

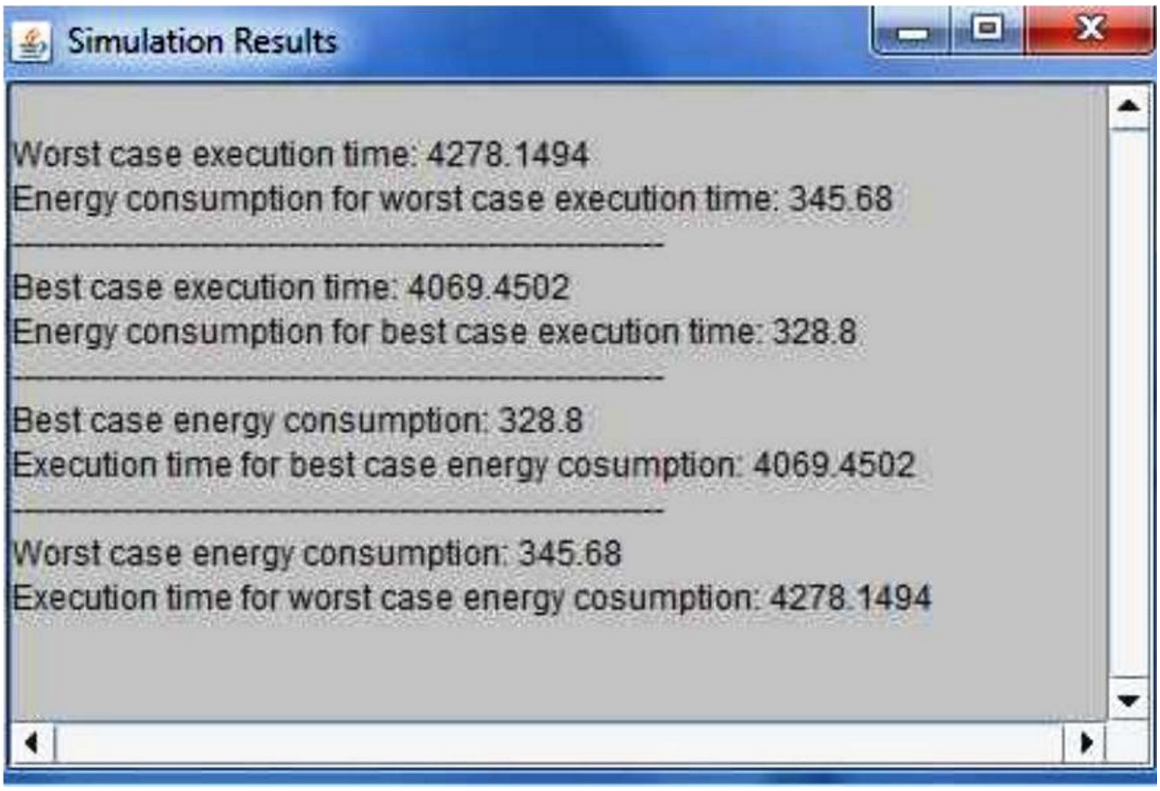

Figura 5.3: Resultados para o controle de rotação

Figura 5.4: Impressoras térmicas

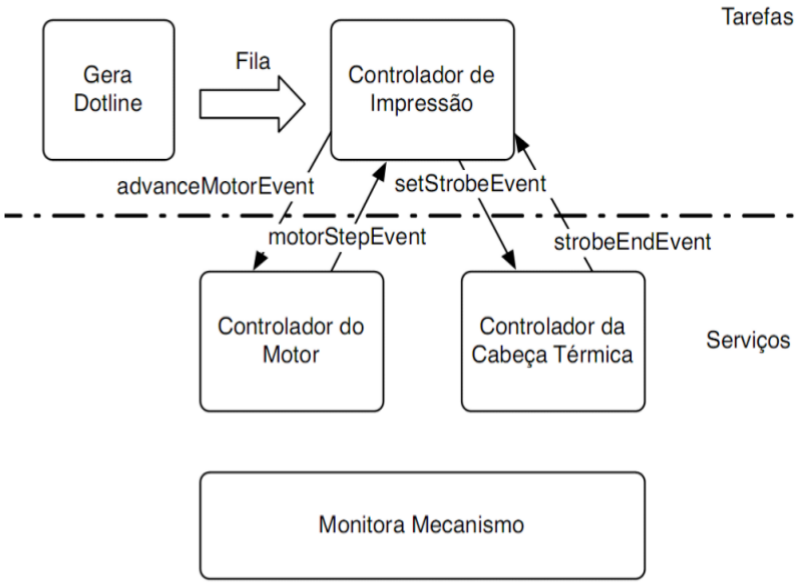

Figura 5.5: Componentes básicos de uma impressora

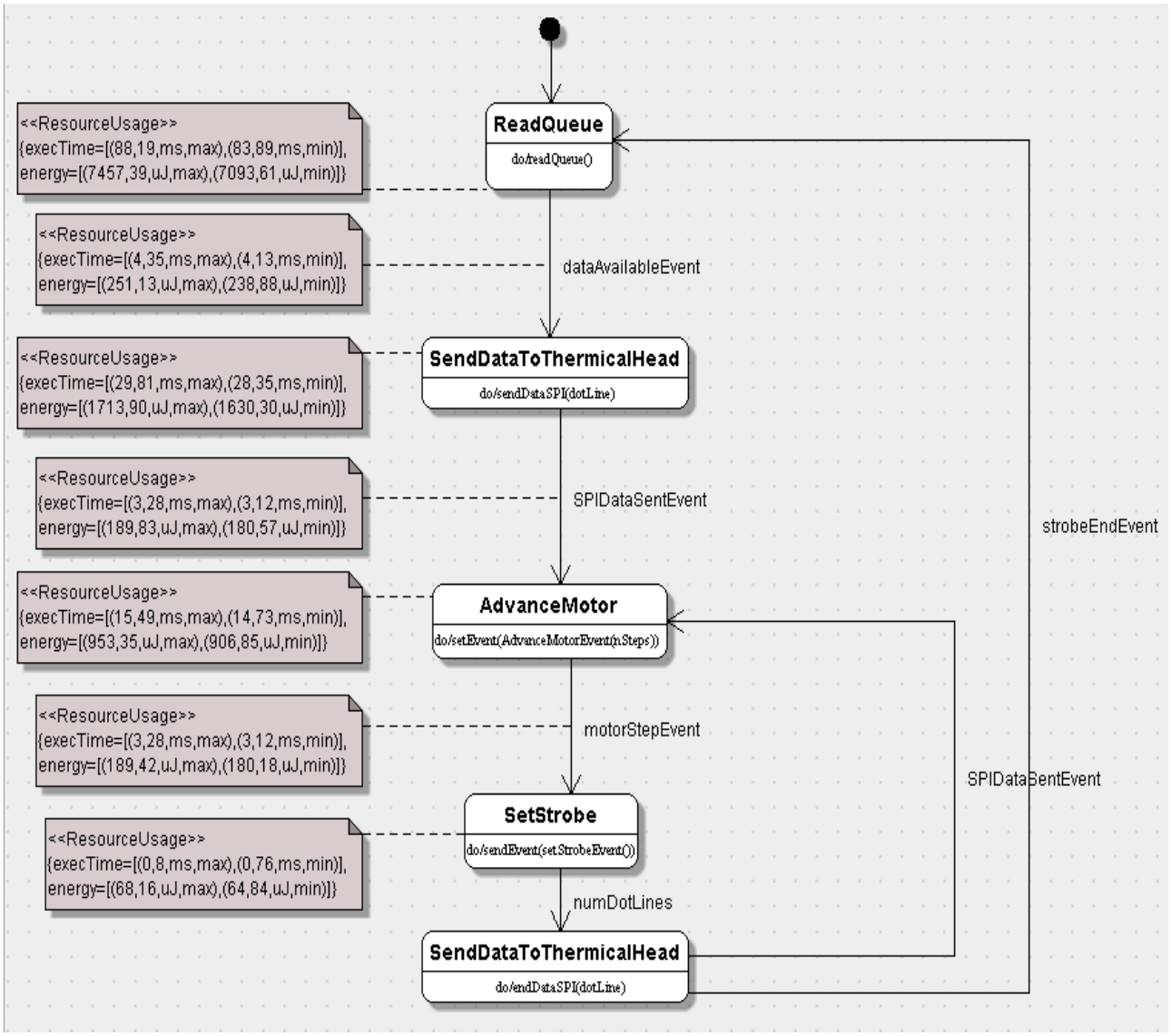

Figura 5.6: Diagrama de estados do controlador de impressão com restrição de tempo e energia

esse sinal for desativado. Se ainda existem linhas de impressão prontas, o processo será repetido, caso contrário, o motor será desligado aguardando que o controlador de impressão receba novamente dados de impressão, recomeçando todo o ciclo. De forma semelhante ao exemplo anterior, o modelo foi obtido automaticamente pela ferramenta Calau. O resultado do mapeamento é apresentado na Figura 5.7.

O modelo ETPN também foi adotado para realizar análises quantitativas. O *BCET* calculado para o controlador de impressão foi 138.1 *ms* e o *WCET* foi 145.2 *ms*. Os resultados obtidos para o melhor e pior caso, referentes ao seu consumo de energia, foram 10295.23 *μJ* e 10823.18 *μJ*, respectivamente. Esses resultados são apresentados na Figura 5.8.

A análise qualitativa do modelo ETPN também foi realizada. Para isso, como mencionado anteriormente, foram gerados arquivos de entrada para a ferramenta INA a partir do modelo ETPN. A análise revelou as seguintes propriedades:

- ***Deadlock Freedom*** - A análise do controlador de impressão indica que o diagrama do controlador de impressão não possui *deadlocks*, ou seja, os estados do controlador de impressão não ficam impedidos de continuar suas execuções à espera de um evento que um outro estado possa executar.
- **Reversibilidade –** O controle de impressão não é reversível para sua marcação inicial, porém possui *home states*, já que o controlador não pode retornar a seu estado inicial após sua inicialização. O estado inicial só pode ser alcançado quando o controlador for iniciado ou reiniciado.
- ***Liveness –*** O modelo do controlador de impressão não *live*, pois existem situações em que as transições são executadas apenas uma vez, isto é, o controlador de impressão só irá executar todos seus eventos quando for inicializado ou reiniciado.

5.3 CONSIDERAÇÕES FINAIS

Através de dois estudos de caso, foram mostradas as principais funcionalidades da ferramenta Calau. É possível realizar automaticamente o mapeamento de um Diagrama de Estados da SysML em uma rede de Petri. Com o modelo formal em mãos, a ferramenta permite realizar análises quantitativas e exportar o modelo para a ferramenta INA, onde são feitas as análises qualitativas.

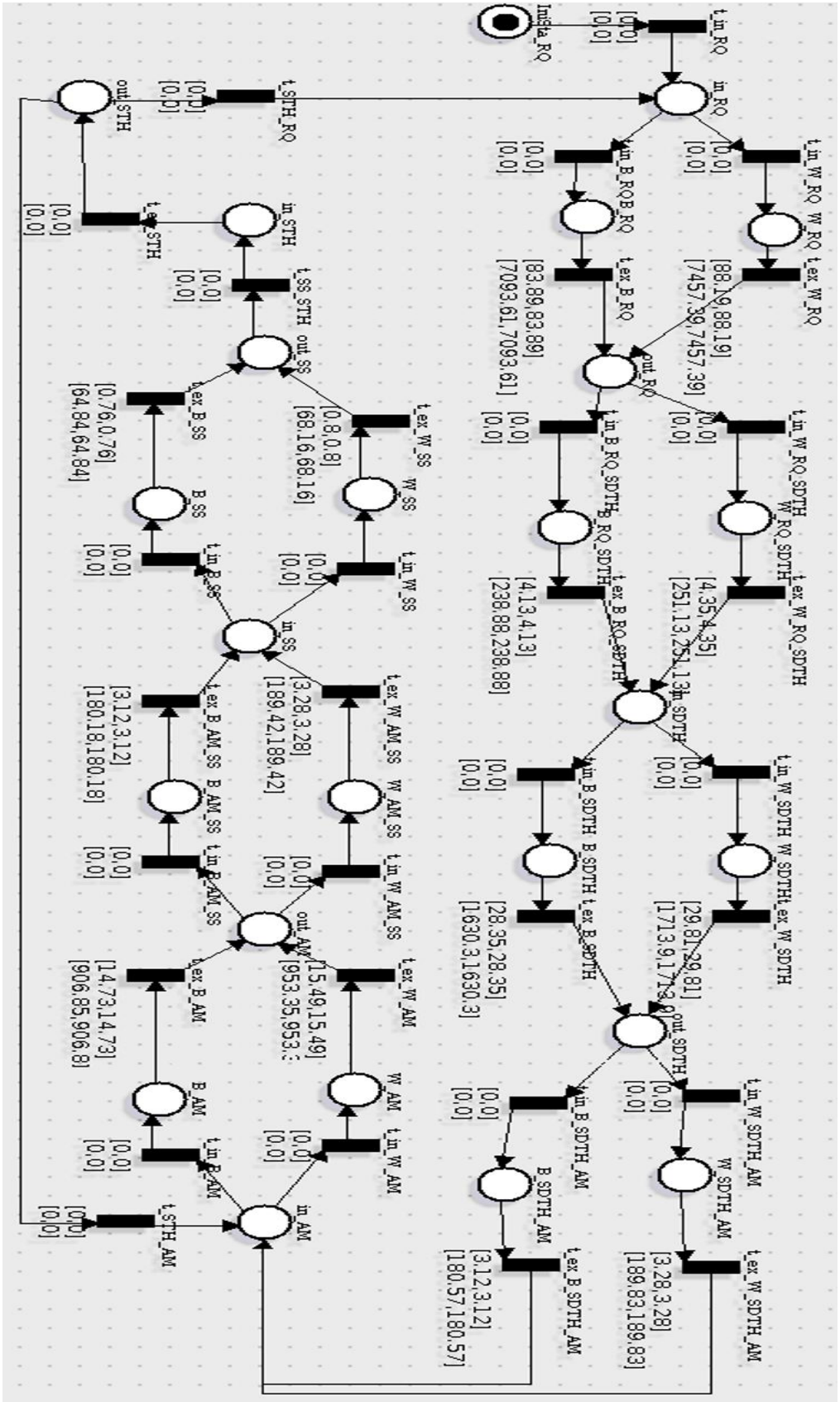

Figura 5.7: Modelo ETPN do controlador de impressão com restrições de tempo e anotações de energia

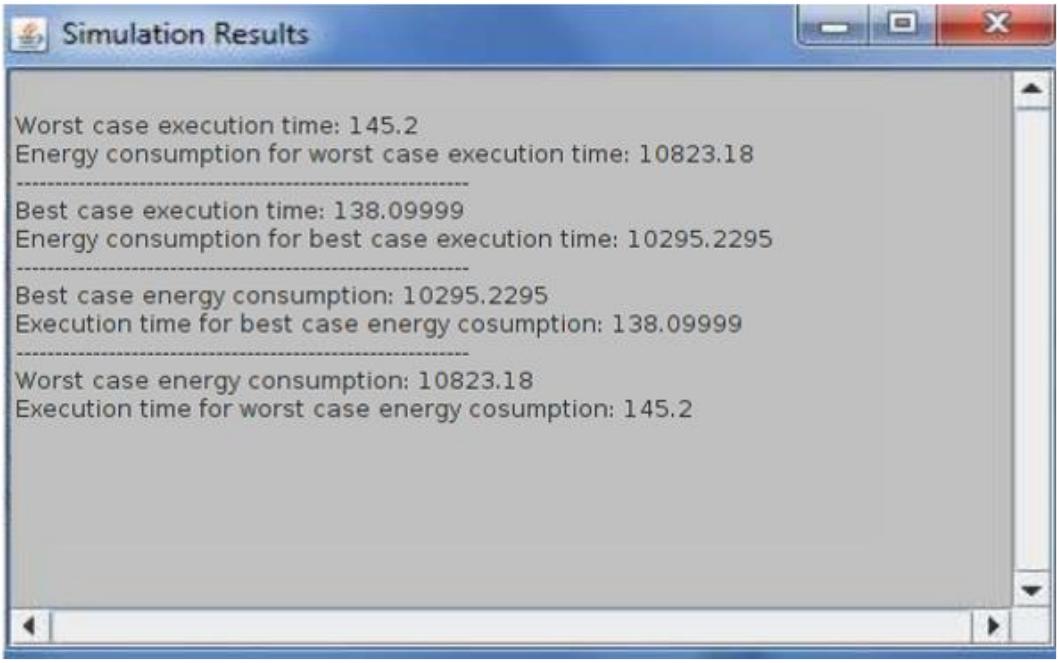

Figura 5.8: Resultados para o controlador de impressão

CAPÍTULO 6

CONCLUSÕES

Com o aumento da complexidade e diversidade dos sistemas embarcados de tempo-real críticos, cresce cada vez mais a necessidade da análise desses sistemas, visto que erros podem ser catastróficos, resultando em perdas de vidas ou de grande quantidade de recursos financeiros. Além disso, a pressão do mercado exige produtividade ao menor tempo e custo possível. Por isso, a adoção de métodos formais na análise de requisitos é uma tarefa fundamental para que os aspectos críticos como, por exemplo, tempo e energia, sejam capturados, precisamente, ainda na fase de especificação do sistema. Isso é possível pelo fato de que modelos formais, que são fundamentados matematicamente, permitem verificações e análises tanto qualitativas quanto quantitativas.

Este trabalho apresentou Calau, uma ferramenta que permite a estimação (tempo de execução e consumo de energia) e verificação de propriedades dos SETR, nas fases iniciais de seu ciclo de desenvolvimento. Para isso, primeiramente, é realizado o mapeamento automático do diagrama de estados da SysML, em um modelo ETPN. As anotações quantitativas do *profile* MARTE, tais como tempo e energia, são levadas em consideração e inclusas no modelo ETPN. Por último, o modelo obtido, que representa o comportamento do diagrama de estados, é adotado para realizar análises e verificações.

Para demonstrar a aplicabilidade da ferramenta, foram realizados dois estudos de caso. O primeiro deles é um controlador de rotação de um motor de automóvel, cuja finalidade é limitar a velocidade do mesmo. O segundo estudo de caso é uma impressora térmica. Esse tipo de impressora produz uma imagem quando a cabeça de impressão térmica passa sobre o papel. Para ambos os estudos, foram apresentadas as análises quantitativas, que dizem respeito ao tempo de execução e ao consumo de energia do sistema modelado. Como também, foram apresentadas as análises qualitativas, que indicam aspectos importantes do sistema.

Como trabalho futuro, a ferramenta poderá ser estendida para cobrir os outros diagramas da SysML, tais como, o diagrama de atividades e o diagrama de sequência, bem como os diagramas comportamentais da UML, como por exemplo, o diagrama de colaboração, diagrama de tempo e diagrama de interatividade. Em especial, esse último é de grande interesse, pois esse diagrama representa a fusão do diagrama de atividades e sequência. Outro trabalho futuro será relacionado a estressar mais a capacidade de simulação e análise das ETPN a fim de prover mais informações aos projetistas. Adicionalmente, a ferramenta poderá ser adaptada à outros tipos de sistemas.

REFERÊNCIAS

[AAM12] Andrade, E., Alves, M. and Maciel, P. (2012). Calau: An Environment for Modeling and Analyzing Embedded Real-Time Systems. *http://www.cin.ufpe.br/ mma2/Calau/*.

[AMCNb] Andrade, E., Maciel, P., Callou, G., and Nogueira, B. (2009). Mapping uml sequence diagram to time petri net for requirement validation of embedded real-time systems with energy constraints. *Proceedings of the 24th Annual ACM Symposium on Applied Computing, 2009.*

[AMCNc] Andrade, E., Maciel, P., Callou, G., and Nogueira, B. (2009). A Methodology for Mapping SysML Activity Diagram to Time Petri Net for Requirement Validation of Embedded Real-Time Systems with Energy Constraints. *Proceedings of the 2009 Third International Conference on Digital Society, 2009.*

[AMCNd] Andrade, E. C., Maciel, P., Callou, G., Nogueira, B. And Araújo, C. (2010). An Approach Based in Petri Net for Requirement Analysis. *Pawel Pawlewski. (Org.). Petri Nets Applications. : INTECH, 2010, v. , p. 1-20.*

[AMF+09] Andrade, E., Maciel, P., Falcão, T., N., Bruno, Araújo, C. and Callou, G. (2009). Performance and energy consumption estimation for commercial off-the-shelf component system design. *Innovations Syst Softw Eng (2010) 6:107–114.*

[AMN+05] Amorim, L., Maciel, P., Nogueira, M, Barreto, R. and Tavares E. (2005). A Methodology for Mapping Live Sequence Chart to Coloured Petri Net. *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, 2005.

[AMN+06] Amorim, L., Maciel, P., Nogueira, M, Barreto, R. and Tavares E. (2006). Mapping live sequence chart to coloured petri nets for analysis and verification of embedded systems. *ACM SIGSOFT Software Engineering Notes*, 31(3):1{25, 2006.

[AMN+10] Andrade, E., Maciel, P., Callou, G., Nogueira, B and Araújo, C. (2009). A COTS-based approach for estimating performance and energy consumption of embedded real-time systems. *Information Processing Letters 110 (2010) 525–534.*

[AND09] Andrade, E. Modelagem e Análise de Especificações de Sistemas Embarcados de Tempo-Real Críticos com Restrições de Energia (2009). Centro de Informática, Universidade Federal de Pernambuco.

[BGdMT98b] Bolch, G., Greiner, S., de Meer, H., and Trivedi, K.S. (1998) Queueing networks and Markov chains: modeling and performance evaluation with computer science applications. *Wiley-Interscience New York, NY, USA, 1998.*

[BP01a] Baresi, L. and Pezzé, M. (2001). Improving UML with Petri nets. *Electronic Notesin Theoretical Computer Science*, 44(4):107-119, 2001.

[BP01b] Baresi, L. and Pezzé, M. (2001). On formalizing uml with high-level petri nets. *Electronic Notesin Theoretical Computer Science*, 276-304, 2001.

[Dav05] David, R. (2005). Discrete, Continuous, And Hybrid Petri Nets. *Springer, 2005.*

[DdSS] Döll, L.M., de Souza, J.U.F., and Stadzisz, P.C. Verificação e validação de Sistemas Orientados a Objetos Usando Redes de Petri.

[DSP05] Distefano, S., and Puliafito, A. (2005). Software performance analysis in uml models. *Techniques, Methodologies and Tools for Performance Evaluation of Complex Systems, 2005. (FIRB-Perf 2005).*

[EES+03] Engblom, J., Ermedahl, A., Sjoedin, M., Gustafsson, J. and Hansson H. (2003). Worst-case execution-time analysis for embedded real-time systems. *International Journal on Software Tools for Technology Transfer (STTT), 4(4):437–455, 2003.*

[GUE04] Guedes, G.T.A. (2004). UML: Uma abordagem prática. *Novatec, São Paulo,2004.*

[GKZH94] German, R., Kelling, C., Zimmermann, A. and Hommel G. (1994). *TimeNET: A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets*. Technische UniversitÄat Berlin, 1994.

[H+87] Harel, D et al. (1987). Statecharts: A Visual Formalism for Complex Systems. 1987.

[HoCU93] Hennessy M. (1993). Timed process algebras: A tutorial. *Program Design Calculi. Proceedings of the NATO Advanced Study Institute. Marktoberdorf, Germany 28 July-9 Aug 1992, M. Broy, Ed., Berlin, 1993, pp. 325–359, Springer Verlag.*

[J98] Junior, M.N.O. (1998). Desenvolvimento de um protótipo para a medida não invasiva da saturação arterial de oxigênio em humanos – Oxímetro de Pulso (*in portuguese*). MSc Thesis, Departamento de Biofísica e Radiologia. Universidade Federal de Pernambuco, 1998.

[Jan98] Janousek, C.V. (1998). Modelling Objects by Petri Nets. PhD thesis, PhD thesis, Department of Computer Science and Engineering, Technical University of Brno, Czech Republic, 1998.

[Jen91] Jensen, K. (1991). Coloured Petri Nets: A High Level Language for System Design and Analysis. *Advances in Petri Nets 1990, 1991.*

[Jen92] Jensen, K. (1992). *Coloured Petri Nets: Basic Concepts, Analysis Methods, and Practical Use*. Springer, 1992.

[KCC+05] Keegan, P., Champenois, L., Crawley, G., Hunt, C. and Webster, C. (2005). Netbeans ide field guide: developing desktop, web, enterprise, and mobile applications. *Prentice Hall Press, 2005.*

[KP00] King, P.and Pooley, R. (2000). Derivation of Petri Net Performance Models from UML Specifications of Communications Software. *Computer Performance Evaluation: Modelling Techniques and Tools: 11th International Conference, TOOLS 2000, Schaumburg, IL, USA, March 27-31, 2000: Proceedings*, 2000.

[KP99] King , P. and Pooley, R. (1999). Using UML to Derive Stochastic Petri Net Models. *Proceedings of the Fifteenth Annual UK Performance Engineering Workshop*, pages 45–56, *1999.*

[LPK+00] Lee, J., Pan, J.I., Kuo, J.Y, Fanjiang, Y.Y and Yang S. (2000). Towards the verification of scenarios with time Petri-nets. *Computer Software and Applications Conference, 2000. COMPSAC 2000. The 24th Annual International,* pages 503{508, 2000.

[MAR07] OMG MARTE. (2007). Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE), *Beta1. 2007.*

[Mar89] Marsan M.A.. (1989). Stochastic Petri Nets: An Elementary Introduction. *Advances in Petri Nets, 424:1–29, 1989.*

[MCBD02a] Merseguer, J., Campos, J., Bernardi, S. and Donatelli S. (2002). A compositional semantics for UML state machines aimed at performance evaluation. *Discrete Event Systems, 2002. Proceedings. Sixth International Workshop on*, pages 295{302, 2002.

[MLC96] Maciel, P., Lins, R. and Cunha, F. (1996). Introdução às Rede de Petri e Aplicações. *X Escola de Computação, Campinas, SP, 1996.*

[MF76] Merlin, P. and Faber, D. (1976). Recoverability of communication protocols: Implicatons of a theoretical study. *IEEE Transactions on Communications, 24(9):1036–1043, Sept. 1976.*

[MKF06] Murphy, G.C., Kersten, M., and Findlater, L.. (2006). How are java software developers using the elipse ide?. *Software, IEEE, 23(4):76–83, 2006.*

[MUR89] Murata, T.. (1989). Petri nets: Properties, analysis and applications. *Proc. IEEE, 77(4):541–580, April 1989.*

[OMG89] The object management group (OMG). (1989). http://www.omg.org/.

[Pet62] C. A. Petri. Kommunikation mit Automaten. PhD Dissertation, Darmstad University, Germany, 1962.

[SR99] Starke, P. and Roch, S. (1999). INA - Integrated Net Analyzer - Version 2.2. Institut für Informatik, Humbolt Universiät zu Berlin.

[Sys07] OMG SysML, Systems Modeling. (2007). Language (SysML)

specification final report. Object Management Group.

[Sin96] Singhal, A. (1996). Real time systems: A survey. Technical report, Computer Science Departmen,. University of Rochester.

[SPT03] OMG SPT. (2003). Profile for Schedulability, Performance, and Time Specification. *Object Management Group*, 2003.

[Tav06] Tavares, E. (2006). A Time Petri Net Based Approach for Software Synthesis in Hard Real-Time Embedded Systems with Multiple Processors. MSc Thesis, Centro de Informática. Universidade Federal de Pernambuco, 2006.

[TMSOJ08] Tavares E., Maciel, P., Silva, B., and Oliveira Jr, M.N. Hard real-time tasks' scheduling considering voltage scaling, precedence and exclusion relations. (2008). *Information Processing Letters, 108(2):50–59, 2008.*

[TJZ07] Trowitzsch, J., Jerzynek, D., and Zimmermann, A. (2007). A toolkit for performability evaluation based on stochastic UML state machines. *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) ICST, Brussels, Belgium, Belgium, 2007.*

[TMS+07] Tavares, E., Maciel, P., Silva, B., Oliveira, M., and Rodrigues,R. (2007). Modelling and scheduling hard real-time biomedical systems with timing and energy constraints. *Electronics Letters, 43(19):1015–1017, 2007.*

[TMSO08] Tavares, E., Maciel, P., Silva, B. and Oliveira, M. (2008). Hard real-time tasks' scheduling considering voltage scaling, precedence and exclusion relations. *Information Processing Letters, 2008.*

[TZ05] Trowitzsch, J. and Zimmermann, A. (2005). Real-Time UML State Machines: An Analysis Approach. *Object Oriented Software Design for Real Time and Embedded Computer Systems, 2005.*

[TZ06] Trowitzsch, J. and Zimmermann, A. (2006). Using UML state machines and petri nets for the quantitative investigation of ETCS. *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools, 2006.*

[TZH05] Trowitzsch, J., Zimmermann, A. and Hommel G. (2005). Towards Quantitative Analysis of Real-Time UML Using Stochastic Petri Nets. *13th Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, 2005.

[UML05] OMG UML. 2.0 Superstructure Specification. *Object Management Group, 2005.*

[VCF+06] Vijaykumar, N.L., Carvalho, S.V., Francês, C.R.L., Abdurahiman, V.

and Amaral, A.S.M. (2006). Performance Evaluation from Statecharts Representation of Complex Systems: Markov Approach. *IV WPerformance, SBC*, pages 183{202, 2006.

[VH99] Vahid, F., and Givargis, T. (1999). Embedded System Design: A Unified Hardware/Software Approach. Department of Computer Science and Engineering, University of California.

[VLM+03] Vinter, R.A., Lisa, W., Michael, L.H. et al. (2003). CPN Tools for Editing, Simulating, and Analysing Coloured Petri Net. *Proceedings of Applications and Theory of Petri Nets*, pages 23{27, 2003.

[Wal88] Walrand, J. (1998). An Introduction to Queueing Networks. *Prentice Hall, 1988.*

ASSINATURAS

_____

Paulo Romero Martins Maciel
(Orientador)

_____

Marcelo Macêdo Alves
(Aluno)