

**INSTITUTO TECNOLÓGICO DE AERONÁUTICA**



**Lucas Lenzi Alves**

**TRANSFORMAÇÃO DE MODELOS SISTÊMICOS PARA  
SIMULAÇÃO DISCRETA**

Trabalho de Graduação  
2021

**Curso de Engenharia Aeroespacial**

**Lucas Lenzi Alves**

**TRANSFORMAÇÃO DE MODELOS SISTÊMICOS PARA  
SIMULAÇÃO DISCRETA**

Orientador

Prof. Dr. Christopher S. Cerqueira (ITA)

**ENGENHARIA AEROESPACIAL**

SÃO JOSÉ DOS CAMPOS  
INSTITUTO TECNOLÓGICO DE AERONÁUTICA

**Dados Internacionais de Catalogação-na-Publicação (CIP)**  
**Divisão de Informação e Documentação**

Lenzi Alves, Lucas

Transformação de modelos sistêmicos para simulação discreta / Lucas Lenzi Alves.  
São José dos Campos, 2021.  
52f.

Trabalho de Graduação – Curso de Engenharia Aeroespacial– Instituto Tecnológico de Aeronáutica, 2021. Orientador: Prof. Dr. Christopher S. Cerqueira.

1. Capella. 2. Arcadia. 3. Sistemas. 4. Simulador. I. Instituto Tecnológico de Aeronáutica.  
II. Título.

## **REFERÊNCIA BIBLIOGRÁFICA**

LENZI ALVES, Lucas. **Transformação de modelos sistêmicos para simulação discreta.** 2021. 52f. Trabalho de Conclusão de Curso (Graduação) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

## **CESSÃO DE DIREITOS**

NOME DO AUTOR: Lucas Lenzi Alves

TÍTULO DO TRABALHO: Transformação de modelos sistêmicos para simulação discreta.

TIPO DO TRABALHO/ANO: Trabalho de Conclusão de Curso (Graduação) / 2021

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

---

Lucas Lenzi Alves  
238, Rua H8B, Apartamento 238  
31400 – São José dos Campos–BR

# TRANSFORMAÇÃO DE MODELOS SISTÊMICOS PARA SIMULAÇÃO DISCRETA

Essa publicação foi aceita como Relatório Final de Trabalho de Graduação



---

Lucas Lenzi Alves

Autor




---

Christopher S. Cerqueira (ITA)

Orientador

**Cristiane  
Aparecida Martins**



Assinado digitalmente por Cristiane Aparecida Martins  
DN: C=BR, OU=Instituto Tecnológico de Aeronáutica,  
O=Cristiane Martins, CN=Cristiane Aparecida Martins,  
E=cmartins@ita.br  
Razão: Eu sou o autor deste documento  
Localização:  
Data: 2021-11-23 14:10:04

---

Prof<sup>a</sup>. Dr<sup>a</sup>. Cristiane Aparecida Martins  
Coordenadora do Curso de Engenharia Aeroespacial

São José dos Campos, 22 de novembro de 2021.

À minha família, aos professores e aos amigos, que me acompanharam e instruíram nessa jornada.

# Agradecimentos

Obrigado, primeiramente, a minha família. Foram sempre os primeiros a me apoiarem e a acreditarem em mim. Reconheço a sorte de estar cercado de pessoas que me amam e me incentivam a realizar meus sonhos, mesmo quando eu próprio duvido deles. Espero conseguir um dia retornar um pouco de toda a força e ajuda que me deram.

Agraço a todos os meus professores que me instruíram e me acompanharam desde o início dos estudos. Principalmente, agradeço ao meu orientador pela incrível paciência, dedicação e suporte nesse último ano. Apesar de todas as dificuldades e momentos críticos, consegui propor soluções e contornar os problemas. Um obrigado especial a coordenadora do curso, por toda ajuda em três anos de curso profissional. Quando problemas apareceram, sempre esteve solícita a me ajudar, a me fazer ter esperanças e seguir em frente.

Ademais, muito obrigado aos meus amigos e meus colegas que moraram comigo ao longo desses oito anos, desde o início de tudo, ao começar a estudar para o vestibular. Quando penso se deveria ter escolhido outra jornada, lembro das relações que construí nesse tempo, e isso torna minhas escolhas ainda mais valiosas. Tenho certeza que não teria chegado tão perto de me formar sem o apoio deles. São inúmeras as horas em que me ajudaram a estudar para provas, trabalhos e exames. Tão importante quanto, obrigado pelas horas de diversão, festas, filmes, conversas sérias e conversas sem sentido. Esses momentos tornaram muito mais leve a faculdade, e levarei as memórias de cinco anos de ITA para a vida toda.

Obrigado a minha namorada, por confiar e me incentivar a me manter resiliente, principalmente numa época que tudo estava mais difícil. Apesar da distância, me pareceu, durante todo esse ano, que estava junto comigo, me empurrando para terminar essa etapa final.

Obrigado ao Lucas de dezessete anos, que saiu de casa e mudou de cidade para estudar e tentar realizar um sonho. Que aprendeu com os erros, as frustrações e reprovações, e não desistiu. Mudou as estratégias, se conheceu e conheceu pessoas incríveis. Por fim, esse é um trabalho com um pouco de cada um que esteve comigo nessa longa, cansativa, mas incrível jornada.







# Resumo

Engenharia Aeroespacial consiste na concepção de sistemas aeroespaciais como satélites, foguetes, espaçonaves, entre outros. São sistemas que integram tecnologias complexas, e necessitam, em sua maioria, de métodos para otimizar os projetos desenvolvidos, com objetivo de minimizar riscos, economizar financeiramente e reduzir o tempo de desenvolvimento. Para isso, a Engenharia de Sistemas torna-se extremamente útil ao ser utilizada em projetos aeroespaciais. Esse campo interdisciplinar de engenharia permite um foco nessa otimização de desenvolvimentos de projetos, podendo atuar em toda a linha do tempo. De maneira resumida, essa engenharia busca relacionar os requisitos do usuário com modelos de soluções possíveis de arquiteturas, realizando processos iterativos para melhorar a solução final. Com o passar dos anos e, em consequência dos avanços tecnológicos, uma metodologia vem sendo utilizada com relevância em projetos complexos: Engenharia de Sistemas Baseada em Modelos (MBSE). O objetivo desse trabalho é de compreender a eficácia e vantagens do uso dessa metodologia no dia a dia do engenheiro de sistemas, e comparar com possíveis usos no setor aeroespacial. A partir da construção de modelos para sistemas do dia a dia, como um elevador, foi possível projetar um sistema básico e entender seu comportamento durante o funcionamento. Foi utilizado software Capella, que utiliza a metodologia Arcadia como solução para aplicação do método MBSE. Além disso, comparou-se com o uso de outros softwares e linguagens de engenharia de sistema, como os diagramas puros em SysML.

# Abstract

Aerospace Engineering is the design of aerospace systems such as satellites, rockets, spacecraft, among others. These are systems that integrate complex technologies, and most need methods to optimize the developed projects, with the objective of minimizing risks, saving financially and reducing development time. For that, Systems Engineering becomes extremely useful when used in aerospace projects. This interdisciplinary field of engineering allows for a focus on this optimization of project development, being able to act across the entire timeline. In short, this engineering seeks to relate user requirements with models of possible architectural solutions, performing iterative processes to improve the final solution. Over the years and as a result of technological advances, a methodology has been used with relevance in complex projects: Model-Based Systems Engineering (MBSE). The objective of this work is to understand the effectiveness and advantages of using this methodology in the systems engineer's daily routine, and compare it with possible uses in the aerospace sector. By building models for everyday systems, such as an elevator, it was possible to design a basic system and understand its behavior during operation. Capella software was used, which uses the Arcadia methodology as a solution for applying the MBSE method. Furthermore, it was compared with the use of other systems engineering software and languages, such as pure diagrams in SysML.

# Lista de Figuras

FIGURA 1.1 – Exemplo de modelagem SysML(MOORE, 2011) . . . . .	18
FIGURA 1.2 – Exemplo de modelagem por Máquina de Estados . . . . .	20
FIGURA 1.3 – Exemplo de máquina de estados simplificada para um satélite . . . . .	21
FIGURA 1.4 – Exemplo de 'zoom' em uma máquina de estados abstrata (HAREL, 1987) . . . . .	22
FIGURA 1.5 – Exemplo de ortogonalidade para o exemplo de máquina de estados de satélite . . . . .	23
FIGURA 2.1 – Metodologia Arcadia(CAPELLA, ) . . . . .	26
FIGURA 2.2 – Catapulta de brinquedo do sistema teste . . . . .	26
FIGURA 2.3 – Catapulta de brinquedo do sistema teste . . . . .	27
FIGURA 2.4 – Catapulta de brinquedo do sistema teste . . . . .	28
FIGURA 3.1 – Diagrama de Sistema e Atores . . . . .	32
FIGURA 3.2 – Diagrama de Capacidades Operacionais . . . . .	32
FIGURA 3.3 – Diagrama operacional do elevador . . . . .	33
FIGURA 3.4 – Diagrama de descrição das atividades operacionais . . . . .	34
FIGURA 3.5 – Diagrama operacional do elevador . . . . .	35
FIGURA 3.6 – Diagrama de arquitetura operacional . . . . .	36
FIGURA 3.7 – Diagrama de contexto do sistema . . . . .	37
FIGURA 3.8 – Diagrama de Missões . . . . .	37
FIGURA 3.9 – Diagrama de Capacidades . . . . .	38
FIGURA 3.10 – Diagrama de intercâmbio do elevador se movendo - parte A . . . . .	39
FIGURA 3.11 – Diagrama de intercâmbio do elevador se movendo . . . . .	40

---

FIGURA 3.12 –Diagrama de arquitetura do elevador se movendo . . . . .	41
FIGURA 3.13 –Diagrama de funcionamento e relações das funções do sistema . . . .	42
FIGURA 3.14 –Cenário em que o elevador está com sobrepeso . . . . .	43
FIGURA 3.15 –Diagrama de arquitetura do elevador em sobre-carga . . . . .	44
FIGURA 3.16 –Arquitetura Lógica . . . . .	45
FIGURA 3.17 –Composição das funções do sistema . . . . .	45
FIGURA 3.18 –Sistema Lógico do Elevador . . . . .	46
FIGURA 3.19 –Máquina de Estados do sistema de movimentação do elevador . . . .	46
FIGURA 3.20 –Código em YAML utilizado para definição da máquina de estado para sistema elevador . . . . .	47
FIGURA 3.21 –Código em python utilizando a bibliotecaismic para exportação do código PlantUML . . . . .	48
FIGURA 3.22 –Código em PlantUML utilizado para visualização da máquina de estado para sistema elevador . . . . .	49
FIGURA 3.23 –Máquina de Estados para Elevador em linguagem SysML . . . . .	49

# Lista de Abreviaturas e Siglas

ITA	Instituto Tecnológico de Aeronáutica
NASA	National Aeronautics and Space Administration
GPS	Global Positioning System
SysML	Systems Modeling Language
MBSE	Model-based systems engineering

# Sumário

1	MOTIVAÇÃO E INTRODUÇÃO TEÓRICA . . . . .	14
1.1	Setor Aeroespacial e a Engenharia de Sistemas . . . . .	14
1.2	Engenharia de Sistemas Baseada em Modelos (MBSE) . . . . .	16
1.3	Máquina de Estados Finita . . . . .	20
1.4	Objetivo . . . . .	24
2	METODOLOGIA . . . . .	25
2.1	Software Capella . . . . .	25
2.2	Primeiro exemplo: catapulta de brinquedo . . . . .	26
2.2.1	Comparação entre Capella e SysML . . . . .	27
2.3	Sismic . . . . .	30
3	RESULTADOS E DISCUSSÕES . . . . .	31
3.1	Utilização do Capella para aplicação de MBSE . . . . .	31
3.1.1	Análise Operacional . . . . .	31
3.1.2	Análise do Sistema . . . . .	36
3.1.3	Arquitetura Lógica . . . . .	44
3.2	Utilização da bibliotecaismic para aplicação de MBSE . . . . .	47
4	CONCLUSÃO . . . . .	50
	REFERÊNCIAS . . . . .	52

# 1 Motivação e Introdução Teórica

## 1.1 Setor Aeroespacial e a Engenharia de Sistemas

Engenharia Aeroespacial é o ramo da engenharia que estuda a “concepção com profundos conhecimentos em projeto e construção de sistemas aeroespaciais, tais como: foguetes, veículos lançadores suborbitais, veículos espaciais e satélites.”(ENGENHARIA... , )

Esses sistemas estão sujeitos as situações críticas, tanto de transporte como de órbita final, como por exemplo, grandes oscilações de temperatura, radiações intensas, vibrações, acelerações, diferenças de pressões, etc. Por conseguinte, são necessárias a implementação e integração de diversas tecnologias complexas. ”Esta complexidade impede um único engenheiro de participar num projeto em todas as suas fases e em vez disso um projeto aeroespacial é levado a cabo por uma equipe de especialistas, cada qual com a sua especialização em determinado ramo da engenharia”.(RODRIGUES, 2013)

Nesse contexto, o conceito de Engenharia Aeroespacial está diretamente atrelado ao conceito de Missão Espacial. Está é definida como o conjunto de parâmetros de missão e refinamento de requisitos necessários para atingir um objetivo determinado em tempo hábil e minimizando o risco e o custo do processo.(WERTZ DAVID F. EVERETT, 2011)

Uma das missões de maior conhecimento público é a primeira viagem do homem a Lua, desenvolvida pelo programa americano Apollo e coordenado pela NASA. Impulsionado pelo contexto histórico da Guerra Fria e, por conseguinte, da Corrida Espacial, as missões desse projeto consistiam em colocar a nave de transporte em órbita, utilizando foguetes Saturno, descer os astronautas na Lua utilizando um módulo lunar e retornar com eles para a Terra, reentrando na atmosfera dentro do módulo de comando e pousando com auxílio de um paraquedas. Todo o projeto envolveu dezenas de engenheiros, físicos, programadores, biólogos, médicos, matemáticos, e várias outras profissões, trabalhando integrados para cumprir o objetivo da missão.

Com o passar das décadas, as missões espaciais evoluíram devido ao desenvolvimento tecnológico (e muito desse desenvolvimento foi consequência das próprias missões espaciais), e os objetivos das missões diversificaram. A necessidade de comunicação rápida e

geolocalização fez com que os satélites artificiais se tornassem as principais cargas úteis das missões. Atualmente, há uma grande diversificação no tamanho, investimento, tecnologia, e outros parâmetros envolvidos. Ainda sim, é necessário uma integração eficiente e consistente entre todos os subsistemas, sejam eles físicos ou lógicos.

De forma geral, pode-se simplificar o sistema aeroespacial em três subsistemas: terrestre, lançador e espacial. O segmento terrestre é aquele, como o próprio nome diz, fixo na Terra. Esse segmento é responsável por receber e enviar dados para o segmento espacial, além de transmitir informações entre si, processar dados, controlar e servir de centro de operações para o restante dos segmentos. O segmento lançador tem como objetivo colocar o satélite em órbita. Esse tipo de transporte é feito pelos veículos lançadores, como foguetes. Existem uma grande diversidade de parâmetros que podem variar de lançador para lançador, dependendo da órbita final da carga útil. Por fim, o segmento espacial é aquele que está em órbita. Normalmente, são satélites artificiais, módulos de transporte espacial, sondas, etc. Esse segmento, normalmente, é responsável por cumprir o objetivo principal da missão. Por exemplo, tirar fotos, enviar dados, obter medições de radiação, etc.

Devido a complexidade desses sistemas, requerem-se ferramentas de exploração e gerenciamento de complexidade para possibilitar aos engenheiros: a visão do domínio do problema identificando as necessidades dos stakeholders e seus requisitos; a visão do domínio da solução objetivando identificar alternativas de arquiteturas que possam vir a se tornar soluções candidatas à concretização do projeto.

Portanto, a engenharia de sistemas desempenha papel fundamental na otimização do projeto, melhorando o desempenho de missão, reduzindo custos e diminuindo riscos de falhas.

Pode-se definir a engenharia de sistemas como "uma abordagem interdisciplinar para traduzir as necessidades dos usuários na definição de um sistema, sua arquitetura e design por meio de um processo iterativo que resulta em uma operação eficaz. A engenharia de sistemas se aplica a todo o ciclo de vida, desde o desenvolvimento do conceito até a disposição final".(MITRE, 2014)

Ademais, um outro conceito que complementa a finalidade da engenharia de sistemas é "uma aproximação multidisciplinar e baseada no bom senso, que permite a realização de um sistema com sucesso".(HOLT, 2018)

O ciclo de vida de um projeto, de maneira geral, pode ser dividido nas etapas:

- Concepção
- Requisitos



- Arquitetura
- Design e Desenvolvimento
- Integração
- Testes
- Operação e Manutenção

A concepção é a primeira etapa do projeto, no qual os stakeholders apresentam o problema e expõem suas necessidades. Em seguida, os engenheiros determinam os requisitos iniciais do projeto, a fim de se cumprir o objetivo da missão. A partir desse momento, é possível desenvolver a arquitetura inicial, coerente com os requisitos. A próxima etapa é o desenvolvimento em si do projeto. Ou seja, transformar as ideias do projeto em um sistema físico e lógico. Normalmente, para facilitar o desenvolvimento do projeto, divide-se ele em vários subsistemas. Portanto, é necessário integrar todas as partes que compõem o todo da missão. Antes de finalizar, são realizados os testes dos sistemas e das interfaces de integração. Por fim, ocorre a operação e, quando possível, a operação do sistema. É necessário destacar que essas etapas não ocorrem necessariamente de maneira linear. A maioria dos projetos ocorrem com várias dessas etapas ocorrendo em paralelo, e também ocorrem diversos processos iterativos, buscando melhoria e resoluções dos problemas que possam surgir. Cabe ao engenheiro de sistemas identificar a melhor metodologia para otimizar durante esse processo não só os custos, mas também o tempo.

Devido aos altos custos envolvidos para o projeto, muitas vezes torna-se inviável o desenvolvimento de vários protótipos do projeto. Além disso, como a missão espacial ocorre fora da atmosfera, essas situações não são simples de simular. Faz-se necessário, portanto, o uso de metodologias matemáticas para simular a situação de operação da missão. Para isso, utiliza-se algumas simplificações, com objetivo de reduzir a necessidade de performance de hardware, além de tornar possível os cálculos considerando a grande quantidade de variáveis envolvidas.

## 1.2 Engenharia de Sistemas Baseada em Modelos (MBSE)

Outro importante conceito a ser estudado é o de Engenharia de Sistemas Baseada em Modelos (MBSE). Esse é um método de Engenharia de Sistemas que consiste na representação de um sistema complexo a partir de simplificações e aproximações, com objetivo de facilitar o estudo do caso real de um projeto. Os modelos buscam representar as características mais importantes e indispensáveis do sistema, de forma que elas descrevam o comportamento esperado para a missão, minimizando ao máximo possível os erros de

projeto. Para compreender o conceito de MBSE, primeiramente faz necessário o aprendizado do conceito de SysML, que é a linguagem de modelagem aplicada a engenharia de sistema.

Essa linguagem surgiu em meio a demanda de otimização dos processos de engenharia modernos, que atingiram um alto nível de complexidade e tecnologia. Pode-se utilizar como exemplo uma tecnologia mais presente no dia a dia: o carro. No início do século XX, o carro era um meio de transporte muito mais simples do que os atuais. Só existia sistema de câmbio manual, utilizavam somente combustíveis fósseis e o sistema elétrico era simples, com principais funcionalidades de dar partida no motor e iluminação. Por outro lado, os projetos de carros atuais atingiram um outro nível de complexidade. Atualmente os carros são providos de sistemas de câmbio automático, motores elétricos, sensores, GPS e computadores de bordo. Inclusive, projetos atuais permitem até que os carros se movimentem em piloto automático. O exemplo do avanço da indústria automotiva expande-se para quase todas as áreas da engenharia, inclusive para o setor aeroespacial.

Sendo assim, ao se adicionar mais funcionalidades a um sistema, não aumentamos somente a quantidade de subsistemas, mas a quantidade de interações e interfaces entre eles. Ademais, cada subsistema pode ser limitado tanto pelos requisitos do sistema como todo, pensando em usuário final, como pelos subsistemas que compõem o todo. "A engenharia de sistemas é uma abordagem que tem sido amplamente aceita na indústria aeroespacial e de defesa para fornecer soluções de sistema para problemas tecnologicamente desafiadores e de missão crítica. Essas soluções geralmente incluem hardware, software, dados, pessoas e instalações". (MOORE, 2011)

"Engenharia de sistemas baseada em modelo (MBSE) é a aplicação formalizada de modelagem para apoiar requisitos de sistema, design, análise, verificação e atividades de validação começando na fase de design conceitual e continuando ao longo do desenvolvimento e fases posteriores do ciclo de vida."(MOORE, 2011)

O modelo do sistema inclui as especificações do sistema, design, análise e verificação. A partir do uso da linguagem SysML, pode-se representar as relações transversais entre todos os elementos do modelo, de forma a possibilitar a identificação de diversas perspectivas do sistema.

A representação simplificada de um modelo genérico pode ser vista na Figura 1.1.

O uso inicial desse método permite o desenvolvimento do design de um sistema de tal forma que satisfaça os requisitos necessários para cumprir os objetivos do projeto. Primeiramente, são desenvolvidas arquiteturas genéricas de software e hardwares para, após análises mais aprofundadas e refinamentos desses requisitos, serem projetadas arquiteturas com softwares e hardwares cada vez mais específicos. O uso da linguagem SysML permite também identificar como cada elemento do projeto está relacionando aos requisi-

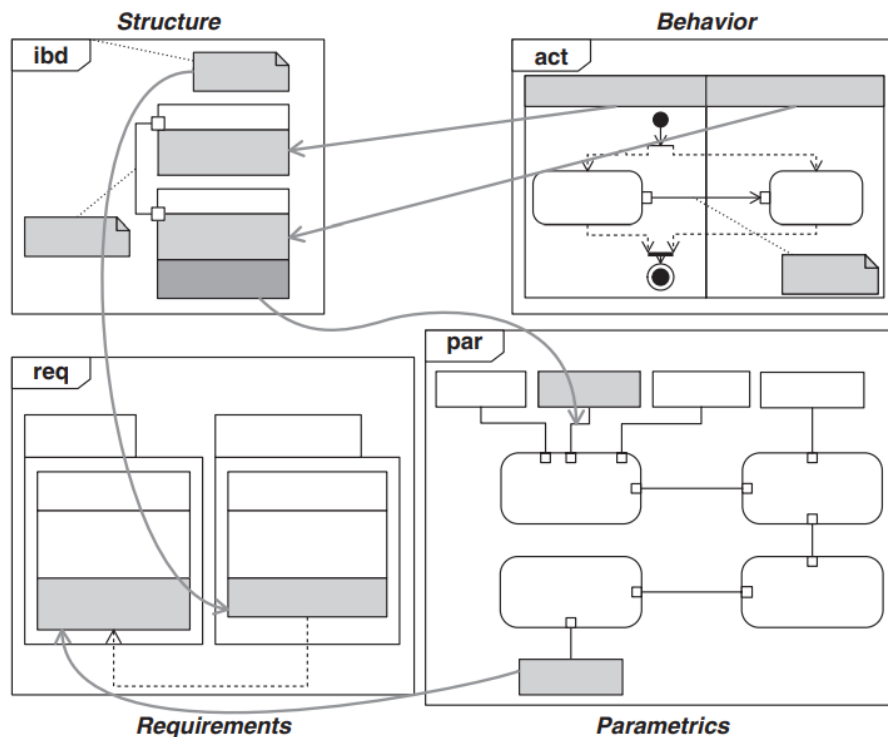


FIGURA 1.1 – Exemplo de modelagem SysML(MOORE, 2011)

tos do sistema. Sendo assim, em caso de mudanças ou atualizações do projeto, é possível ter uma visão macro do impacto sobre o projeto.

Ademais, o modelo atribuído ao projeto pode ser utilizado em análises e simulações complementares, para identificar como o sistema reage para cada situação possível dentro do universo de operação proposto.

A modelagem do sistema utiliza elementos gráficos para representar cada componente do sistema, além das relações entre esses componentes. Sendo assim, são desenvolvidos diagramas que representam as conexões entre requisitos necessários para cumprir-se as missões com os componentes que satisfazem esses requisitos, considerando os limites impostos no projeto.

Por exemplo, na construção de um lançador de satélites, é necessário um motor foguete para impulsionar o lançador para a órbita desejada. Todavia, esse motor, além de possuir empuxo suficiente para tal feito (determinado pelos requisitos de órbita), está limitado pelos requisitos dos demais subsistemas, como o limite físico para as dimensões desse motor.

Além disso, as ferramentas utilizadas para aplicação do método MBSE possibilitam obter documentos gerados automaticamente. Esse documentos tem função similar ao tradicional método de documentação utilizado em projetos mais antigos. A análise desses documentos permite otimizar os custos de desenvolvimento e manutenção do projeto, visto

que é possível um acompanhamento detalhado de cada elemento que compõem o sistema.

Por fim, a partir da elaboração do modelo, é possível executar verificações de rotina que analisam possíveis violações dos requisitos necessários para cumprimento dos objetivos, identificando necessidades de modificações do projeto para contornar problemas de desenvolvimento, identificando quais arquiteturas que vão de encontro às restrições impostas, seja pelo projetista, seja pelas condições de funcionamento do projeto.

Em resumo, segundo (MOORE, 2011), pode-se identificar as principais vantagens do uso de engenharia de sistemas baseado em modelos, em contrasta ao uso do método tradicional baseado em documentação:

- Comunicações aprimoradas:

Compreensão compartilhada do sistema entre a equipe de desenvolvimento e outras partes interessadas; Capacidade de apresentar e integrar visões do sistema de múltiplas perspectivas.

- Redução dos riscos ao longo do desenvolvimento:

Validação contínua de requisitos e verificação de projeto; Estimativas de custos mais precisas para desenvolver o sistema.

- Melhoria da qualidade final do sistema:

Requisitos mais completos, inequívocos e verificáveis; Rastreabilidade mais rigorosa entre requisitos, projeto, análise e teste; Integridade de design aprimorada.

- Produtividade aumentada

Análise de impacto dos requisitos ocorre de maneira mais rápida e abrangente, além de maior velocidade na alterações do design; Exploração mais eficaz das arquiteturas e interfaces; Reutilização de modelos existentes para a aprimoramento do design.; Minimização dos erros e do tempo durante a integração e testes; Geração automatizada de documentos.

- Possibilidade de aproveitamento dos modelos durante as fases de ciclo de vida:

Apoia-se o treinamento do operador sobre o uso do sistema.; Suporte a diagnósticos e manutenção do sistema; Transferência de conhecimento aprimorada; Captura eficiente de conhecimento de domínio sobre o sistema de uma forma padronizada que pode ser acessado, consultado, analisado, desenvolvido e reutilizado.

Em contrapartida, o uso do MBSE traz aos engenheiros novos desafios de adaptação e aprimoramento dos conhecimentos sobre engenharia de sistemas. É necessário treinamentos e estudos dedicados a esse nicho de conhecimento. O objetivo desse trabalho consiste,

entre outras coisas, em verificar a possibilidade e eficiência do uso de conhecimentos de MBSE para a Engenharia Aeroespacial, apontando os pontos positivos e negativos dessa implementação.

### 1.3 Máquina de Estados Finita

Uma dos métodos mais utilizados é a Máquina de Estado, também chamada de autômato. Esse é um método de modelagem matemática que representa um problema complexo a partir de um diagrama de estados ou de uma tabela de transição.

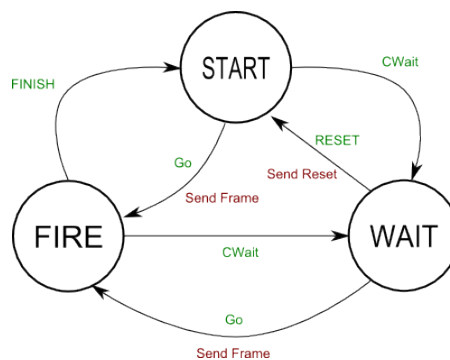


FIGURA 1.2 – Exemplo de modelagem por Máquina de Estados

O estado é um conjunto único de propriedades que definem a situação do sistema. A mudança das propriedades desse sistema são as transições, que acontecem sob condições específicas, determinadas pelo modelador. Um exemplo de como é um desses diagramas é mostrado na Figura 2.1

Uma transição de estados pode tomar, a princípio, a seguinte forma: "quando o evento 'alpha' ocorre durante o estado A, e as condições C são verdadeiras nesse instante, o sistema transfere para o estado B.

Conforme mostrado, para essa modelagem, existem três estados possíveis para o sistema: 'Início', 'Espera' e 'Fogo'; em tradução livre. As transições possíveis estão representadas pelas flechas, informando o requisito necessário para ocorrerem.

Para um sistema de satélite, podemos exemplificar de forma parecida. Tomemos, de maneira simplificada, um satélite em órbita baixa, que tem como objetivo fotografar somente a região do Brasil e enviar as fotografias para a estação de controle em terra. Podemos identificar três estados básicos que esse sistema terá para cumprir seu objetivo: fotografando, enviando dados e standby. Esse último, seria o estado em que o satélite está apenas recarregando suas baterias, e com o mínimo de funcionalidades ativas, para economizar potência consumida. Podemos definir as transações que ocorrem entre esses estados: o satélite só irá fotografar após estar região de órbita desejada, ou seja, passando

pelo Brasil. Após fotografar, ele irá enviar os dados após salvar essas imagens. Por fim, após enviar os dados e sair da região do Brasil, o satélite voltará a situação de standby. Após o engenheiro identificar quais são os estados de funcionamento do satélite e as condições de transição, é possível construir o diagrama no qual visualizamos essa máquina de estados, conforme Figura 1.3.

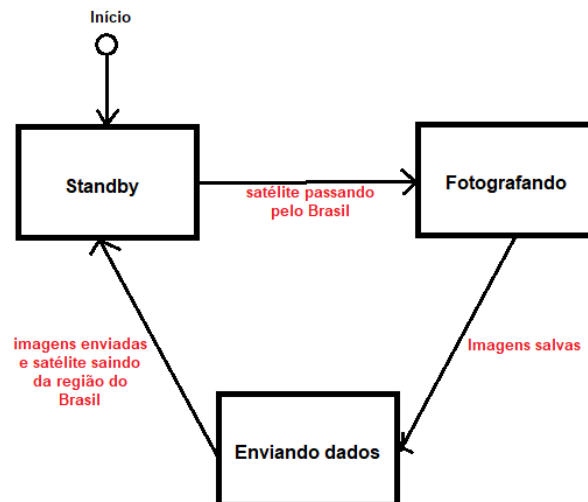


FIGURA 1.3 – Exemplo de máquina de estados simplificada para um satélite

É preciso reforçar que esse é um modelo extremamente simplificado, utilizado para estudo teórico das máquinas de estado. É possível aumentar a complexidade dos elementos desse sistema, e discretizar ainda mais os estados e transições de cada função do satélite. Por exemplo, em todos os momentos desse ciclo, ocorrem correções de atitude de órbita. Essas correções ocorrem em paralelo a várias outras funcionalidades do sistema e poderia ser representada na máquina de estados. É preciso que o engenheiro identifique o melhor ponto de modelagem para que o sistema tenha a complexidade suficiente para os fins a que o modelo se destina, mas que também não tenha complexidade em excesso, o que poderia tornar a análise mais demorada e menos otimizada.

Sendo assim, uma máquina de estados pode ser considerada uma tentativa de analisar modelos complexos e compreender o fluxo de funcionamento do sistema. "Elas constituem uma formalidade gráfica para descrever estados e transições de forma modular, permitindo agrupamento, ortogonalidade (ou seja, simultaneidade) e refinamento, além de incentivo ao 'zoom', ou seja, um recurso para mover-se facilmente para frente e para trás entre os níveis de abstração."(HAREL, 1987)

Com objetivo de tornar as informações mais organizadas e otimizar o uso do espaço do diagrama, algumas padronizações de representações gráficas são adotadas. Para cada estado, utiliza-se blocos retangulares para representar os estados do sistema. Um estado pode ser representado decomposto em dois ou mais sub-estados. Para representar essa

interação, utiliza-se encapsulamento dos retângulos. Ou seja, o macro-estado é representado por um retângulo maior, enquanto seus sub-estados são representados por retângulos menores no interior do maior. Para representar as transições de estados, são utilizadas setas indicando o estado de origem e o estado final, com informações sobre as condições necessárias para ocorrer a mudança entre eles. Essas informações são chamadas de "eventos". No exemplo anterior do satélite, "imagens salvas" é um evento necessário para que ocorra a transição do estado "Fotografando" para "Enviando dados".

Para exemplificar o agrupamento de sub-estados em um macro-estado maior, podemos observar a Figura 1.4.

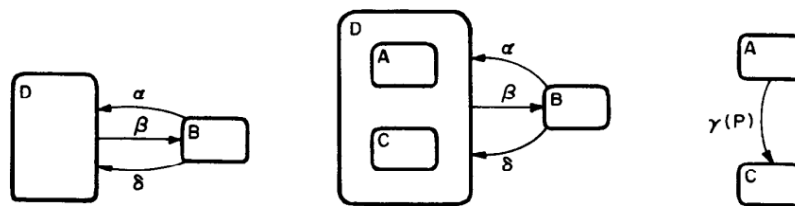


FIGURA 1.4 – Exemplo de 'zoom' em uma máquina de estados abstrata (HAREL, 1987)

Partindo a imagem do meio, pode-se observar os sub-estados do estado 'D', conforme mostrado no diagrama a direita. Essa análise é um "zoom" de aproximação, em que podemos observar os sub-estados que compõem 'D': 'A' e 'C'. Além disso, temos uma única transição nessa visão, dada por  $\gamma(P)$ .

Também é possível ter uma visão ainda mais macro, em que não representamos os sub-estados de 'D', conforme a figura da esquerda. Essa visão simplifica o diagrama completo, e permite representar somente as transições entre 'D' e 'B'.

É preciso reforçar que nenhuma dessas visões é "mais correta" do que as demais. É função de um bom engenheiro de sistemas identificar qual diagrama apresenta melhor utilidade para cumprir os objetivos do modelo proposto.

Outro conceito importante para Máquinas de Estado é o de ortogonalidade. Tomemos o exemplo do satélite citado. Conforme citado, enquanto o satélite desempenha sua função principal de capturar imagens, existem outras funções sendo executadas em paralelo, de maneira independente. Por exemplo, conforme mostrado na Figura 1.5, podemos visualizar no diagrama o controle de atitude do satélite, atuando de maneira independente aos estados relativos a fotografia. Isso ocorre pois é necessário que o satélite esteja sempre na órbita correta, de modo que os atuadores possam corrigir pequenas discrepâncias entre a atitude desejada e a atual. Se um satélite dependesse de um dos estados de fotografia para atuar ou não no controle de atitude, poderiam acumular erros de atitude tais quais os atuadores não conseguissem operar de maneira satisfatória.

Por fim, é possível observar como o encapsulamento de sub-estados pode ser represen-

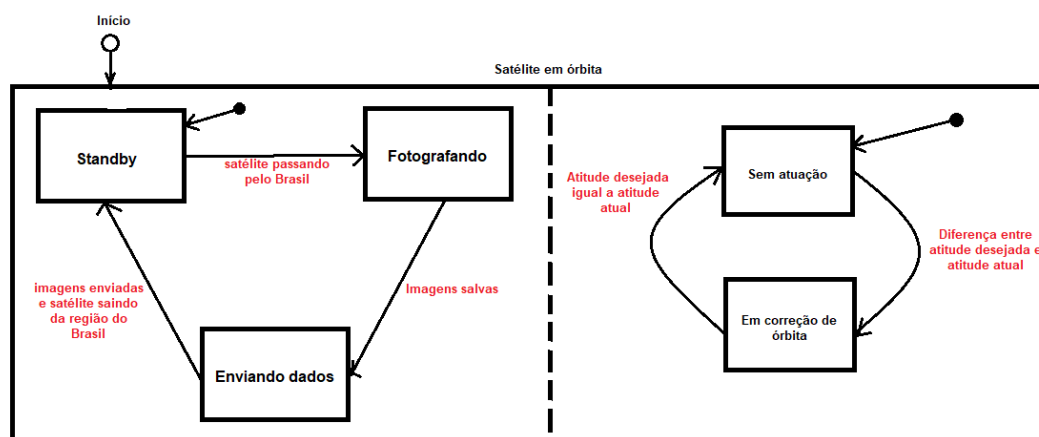


FIGURA 1.5 – Exemplo de ortogonalidade para o exemplo de máquina de estados de satélite

tado. Para esse exemplo, temos o macro-estado satélite em órbita, e cinco sub-estados pertencentes a ele, sendo que três ocorre de maneira ortogonal aos outros dois.

Conforme o modelo vai se aproximando de um caso real de projeto, a complexidade dos diagramas vai aumentando consideravelmente, por isso destaca-se a importância do uso correto de hierarquias de estados, com objetivo de simplificar as visões, quando possível.

O desenvolvimento do formalismo das Máquinas de Estado ocorreu durante um projeto de um sistema aeronáutico de última geração, desenvolvido pela Israel Aircraft Industries (IAI) (HAREL, 1987). Esse tipo de projeto necessita de uma grande colaboração entre os diversos profissionais envolvidos: como engenheiros de software, hardware, comunicação, especialistas em defesa e armamentos, entre outros. É necessário que todos os subsistemas estejam integrados e otimizados para bom funcionamento do avião, reduzindo os riscos de problemas. Tal projeto é aprimorado diversas vezes durante seu ciclo de desenvolvimento. Os requisitos são revisados e testes são feitos para verificar se as novas soluções implementadas solucionam essas necessidades.

Quando comparamos o desenvolvimento de um avião de alta tecnologia com o setor aeroespacial, é possível apontar muitas semelhanças entre os projetos. Um satélite, por exemplo, necessita dessa integrações entre vários subsistemas. Por conseguinte, como a formalização dessa metodologia ocorreu em um contexto similar ao aeroespacial, podemos considerar uma perspectiva de resultados positivos do uso de Máquina de Estados para esses sistemas. Por fim, resultados como otimização do período de rotatividade de testes e melhoria na descrição comportamental do sistema foram apresentados devido a implementação desses diagramas (HAREL, 1987).



## 1.4 Objetivo

Sendo assim, o objetivo desse trabalho é estudar a viabilidade da aplicação da metodologia MBSE como solução de engenharia de sistemas para projetos complexos, analisando alguns exemplos de softwares utilizados para isso, como Capella e biblioteca Sismic, e comparar as vantagens do uso de cada um deles.

## 2 Metodologia

Esse trabalho desenvolve um exemplo de sistema utilizando, primeiramente, o software Capella, com objetivo de obter a máquina de estados finita. Em seguida, utiliza-se a biblioteca `sismic` da linguagem Python para construção da máquina de estados equivalente, para que, dessa forma, seja possível comparar o processo e os resultados utilizando MBSE a partir desses softwares.

### 2.1 Software Capella

O Capella é um software de modelagem de sistemas, baseado na linguagem SysML. Essa linguagem permite uma ampla integração com demais softwares de engenharia. Ademais, essa linguagem utiliza quatro principais pilares como base:

- Estrutura
- Comportamento
- Requisitos
- Parâmetros

O Capella utiliza a metodologia de processos denominada de Arcadia. Esse método divide o processo em quatro partes, duas de análise e duas de arquitetura:(CAPELLA, )

- Análise Estrutural
- Análise de Sistemas
- Arquitetura Lógica
- Arquitetura Física

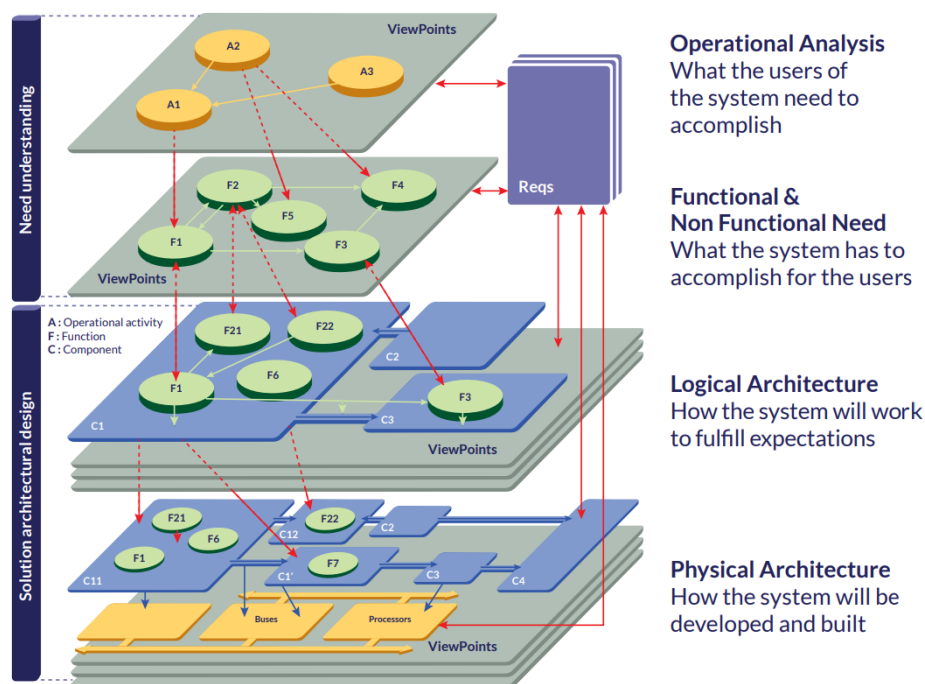


FIGURA 2.1 – Metodologia Arcadia(CAPELLA, )

## 2.2 Primeiro exemplo: catapulta de brinquedo

Os primeiros passos do trabalho consiste na familiarização com todos os softwares envolvidos.

Um primeiro sistema teste consiste em uma catapulta de brinquedo, como mostrado na Figura 2.2, e duas pessoas: a criança e seu pai. As relações de interações entre elas são mostradas na Figura 2.3.



FIGURA 2.2 – Catapulta de brinquedo do sistema teste

Essa imagem representa a análise operacional da metodologia Arcadia. Conforme visto na imagem, a criança tem ação de retirar o brinquedo da caixa, brincar, transferir para o pai ou abandonar o objeto no local onde brinca. Já o pai pode retornar o brinquedo para a caixa ou ensinar a criança a usar o brinquedo.

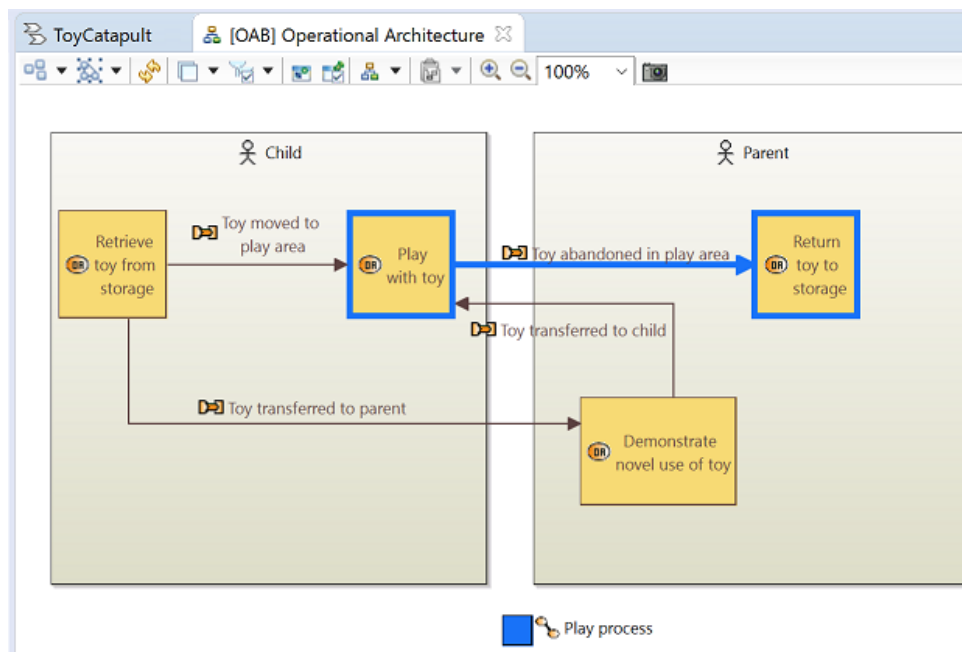


FIGURA 2.3 – Catapulta de brinquedo do sistema teste

Outro resultado para esse teste é o diagrama da arquitetura lógica do sistema, conforme mostrado na Figura 2.4

A arquitetura lógica do sistema mostra todo o processo com mais detalhes, levando em conta as interações e trocas de fluxos entre os blocos. Para o Capella, os fluxos podem ser de massa e energia, por exemplo.

Nesse trabalho, será utilizada a mesma metodologia, todavia, para criação de um novo modelo do zero: o sistema de um elevador com capacidade para 6 passageiros. Utilizando os conceitos teóricos apresentados e criando a modelagem desse sistema, será possível identificar as vantagens e desvantagens do uso do software Capella como solução para aplicação de MBSE. Todavia, para compreender melhor essa metodologia, é necessário identificar as semelhanças e diferenças entre Capella e linguagem SysML.

### 2.2.1 Comparação entre Capella e SysML

Primeiramente, é preciso enfatizar que a comparação entre um software e uma linguagem de engenharia de sistemas pode parecer estranha a princípio. Todavia, o objetivo da comparação é entender o contexto em que essas ferramentas são utilizadas.

SysML é uma extensão da Linguagem Unificada de Modelagem (UML), ou seja, uma linguagem padronizada para elaboração de diagramas. É uma ferramenta genérica, que não apresenta um método para modelagem, mas sim é compatível com várias soluções disponíveis em forma de software, por exemplo. Por ser uma linguagem, apresenta um espectro largo de aplicações na engenharia. Por outro lado, o Capella é um software que

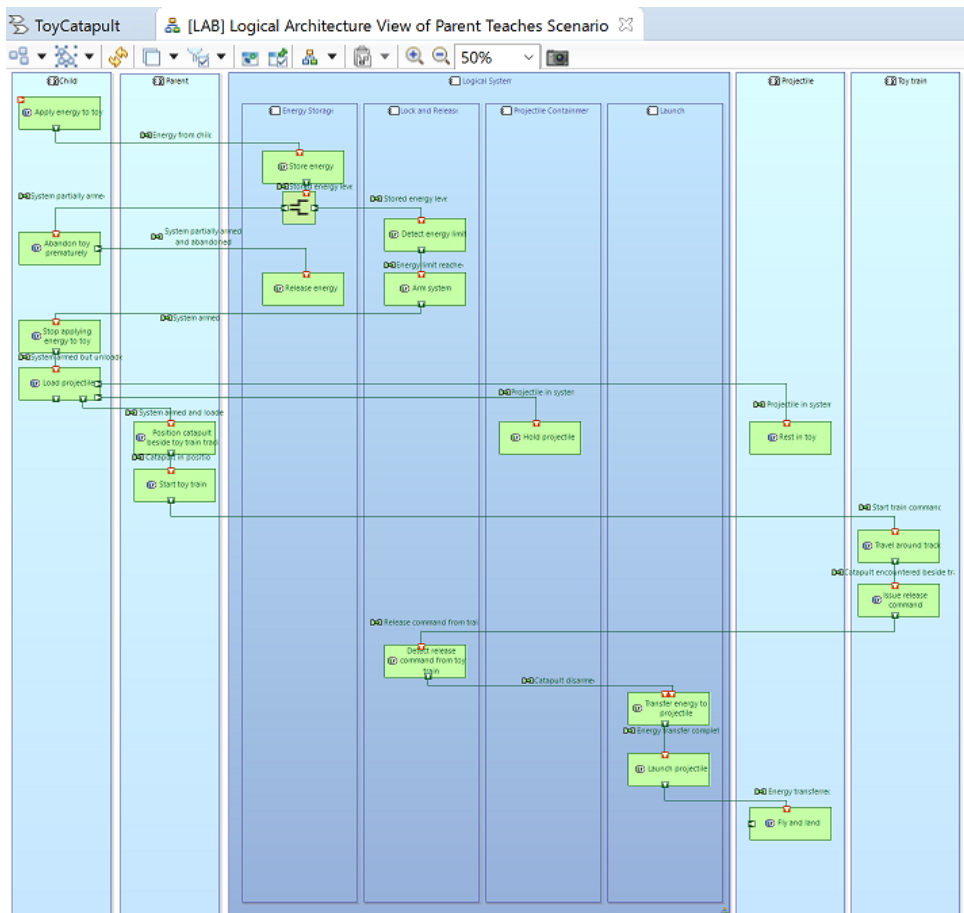


FIGURA 2.4 – Catapulta de brinquedo do sistema teste

utiliza uma metodologia explícita, e já citada: o Arcadia. Esse método reforça a separação entre o contexto de requerimentos para solução do problema e a solução em si, ou seja, a arquitetura genérica que irá satisfazer todos os requisitos impostos. O Capella utiliza o SysML como inspiração para linguagem. Sendo assim, em sua maioria, as representações dos diagramas nesse software se assemelha em muito a linguagem SysML, mas em alguns pontos apresenta algumas diferenças.

De acordo com (EQUIVALENCES...), para o Diagrama de Blocos, SysML utiliza caixas retangulares para definir os elementos do sistema, com suas propriedades e operações. Além disso, utiliza setas e símbolos para representar as relações entre esses elementos. Já no Capella, os blocos são denominados componentes. Não são explícitas suas propriedades, e as relações são mostradas em fluxo de composição em árvore ou por interfaces de componentes, onde pode ocorrer encapsulamento de blocos para representar um elemento contido em outro. A linguagem SysML utiliza encapsulamento de blocos também para representar elementos contidos em outros. Todavia, esses encapsulamentos são utilizados no Diagrama de Blocos Interno. No software, o equivalente a esse diagrama é o de Arquitetura. Nele, são apresentados os componentes em termos de conexões e composições.

O Diagrama de Atividade representa, em SysML, um fluxo de ações que são desenca-

deadas por atividades. Para representar esse ciclo de funcionamento, o Capella utiliza o conceito de 'função'. O software permite alocar funções para os componentes do sistema, e representar o desencadear das funções por meio de gatilhos, ou seja, atividades que acionam tais funções em cada componente.

Em SysML, o Diagrama de Sequências tem como foco visualizar as interações e ações dos componentes em uma linha do tempo. É uma visualização que permite compreender a sequência de ações realizadas pelos elementos e as relações que desencadeiam essas sequências. Da mesma maneira, o Capella utiliza uma visualização extremamente parecida, com diferença na maior variedade de elementos que podem ser referenciados nessa sequência, principalmente pela visualização das funções criadas no diagrama de atividade diretamente nessa visão.

Tanto em SysML quanto no Capella, o Diagrama de Máquina de Estados representa em seus blocos os estados de funcionamento do sistema, as ações e, por conseguinte, as transições entre esse estados, explicitando os eventos que deram origem as mudanças. O destaque para esse diagrama é a simplificação da criação do diagrama de Máquina de Estados, a partir do momento que os diagramas anteriores são criados e as funções são estabelecidas. Além disso, o software apresenta uma ferramenta de uso dos símbolos de diagrama que permite ao engenheiro adicionar estados, regiões, inícios, transições de maneira direta, simples e organizada ao diagrama. Existe a contrapartida de que, em troca dessa simplificação, o Capella não apresenta a mesma capacidade de expressividade da linguagem pura em SysML. Mas como o software visa facilitar o desenvolvimento do projeto com Engenharia de Sistemas, e não necessariamente uma modelagem em baixo nível para representar com excesso de detalhes os subsistemas, essa perda é compatível com os objetivos do uso do software.

Outro diagrama representado em SysML é denominado de Caso de Uso. Ele representa os principais atores e suas relações com o sistema, apresentando as funções descritas em alto-nível. O Capella utiliza o mesmo conceito, todavia introduz a nomenclatura 'Capacidade' para representar as funções de alto nível que o sistema deve prover.

Os requisitos do sistema podem ser apresentados no Diagrama de Requisitos em SysML. Já no software, eles podem ser apresentados dentro dos demais diagramas, como o de sequência e de arquitetura, explicitando suas relações com os demais elementos.

Enquanto SysML utiliza diagramas específicos para representar os parâmetros do sistema, o Capella permite acrescentar os parâmetros (por meio de equações) como propriedade dos componentes dos demais diagramas. Percebe-se mais uma vez uma simplificação do modelo, minimizando o número de diagramas totais e possibilitando uma melhor visualização do contexto dos parâmetros dentro do sistema.

## 2.3 Sismic

”Sismic é uma biblioteca de máquina de estados utilizada para linguagem python, que fornece uma série de ferramentas para definir, validar, simular, executar e testar máquinas de estado”(SISMIC... , ). Foi desenvolvida pelo Dr. Alexandre Decan, da Universidade de Mons.

A definição das máquinas de estado utiliza a ferramenta YAML. Essa é ”uma linguagem de transformação de dados (...) amigável para humanos, projetada em torno dos tipos de dados nativos comuns de linguagens de programação dinâmicas. É amplamente útil para necessidades de programação que variam de arquivos de configuração a mensagens na Internet, persistência de objetos, auditoria e visualização de dados.”(YAML... , ) No entanto, para visualização da máquina de estados com simbologia UML, é necessário utilizar o software PlantUML, que permite a transformação da descrição YAML em diagramas explícitos. Sendo assim, para construção da máquina de estados, pode-se utilizar o código em YAML importado para o python, ou utilizar funções da própria biblioteca para definir e/ou modificar a máquina de estados. Em seguida, é possível exportar o código para ser utilizado no PlantUML, utilizando a função ”export\_to\_plantuml()”dessa biblioteca.

## **3 Resultados e Discussões**

### **3.1 Utilização do Capella para aplicação de MBSE**

Para analisar e comparar o uso da engenharia baseada em modelos, primeiramente, foi obtido como resultado os diagramas de operação, arquitetura e lógica do sistema de um elevador. Para esse trabalho, não foi considerado pertinente a análise da estrutura física dos componentes do sistema, visto que deseja-se comparar a aplicação do modelo de Máquinas de Estado e sua eficácia para engenharia de sistemas.

#### **3.1.1 Análise Operacional**

O primeiro diagrama obtido foi o de sistema e atores. Vamos partir do problema de desenvolvimento do projeto desse elevador. A princípio, temos como ator a pessoa que irá entrar e ser transportada pelo elevador. O sistema será o próprio elevador. Foi criado um sistema genérico 'Transporte', e um sistema específico, 'Elevador'. Esse passo não seria necessário para o modelo proposto. Todavia, foi utilizado para teste das ferramentas disponíveis pelo Capella. Fazendo a comparação com o projeto aeroespacial, poderia ser utilizado 'satélite' como um sistema genérico, e 'satélite de obtenção de imagens' para um sistema específico. O sistema estar contido em outro significa que o esse sistema apresenta todas as propriedades do sistema genérico, mas com objetivo de obter um resultado específico para a missão. O diagrama pode ser visualizado na Figura 3.1.



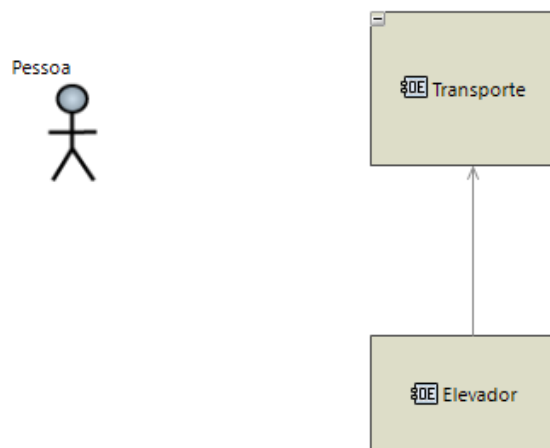


FIGURA 3.1 – Diagrama de Sistema e Atores

Em seguida, foi obtido o diagrama que representa a capacidade operacional do sistema. Ou seja, o que o sistema deve ser capaz de fazer, e com quais atores ele interage. Nesse caso, o elevador irá transportar a pessoa entre andares, conforme Figura 3.2.

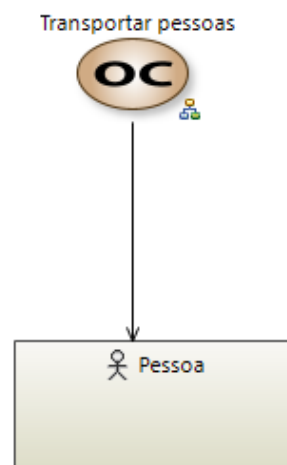


FIGURA 3.2 – Diagrama de Capacidades Operacionais

Então foi possível criar então o diagrama para descrever as atividades operacionais, e os gatilhos para ocorrer cada uma dessas atividades. Para esse diagrama, o foco é desenvolver de maneira genérica as interações de sistema e atores. Não é necessário (e nem interessante) desenvolver arquiteturas de solução até agora. O método Arcadia divide a parte operacional de arquitetura justamente para não limitarmos nosso sistema a soluções específicas logo de início, o que prejudica a organização e otimização da engenharia de sistemas.

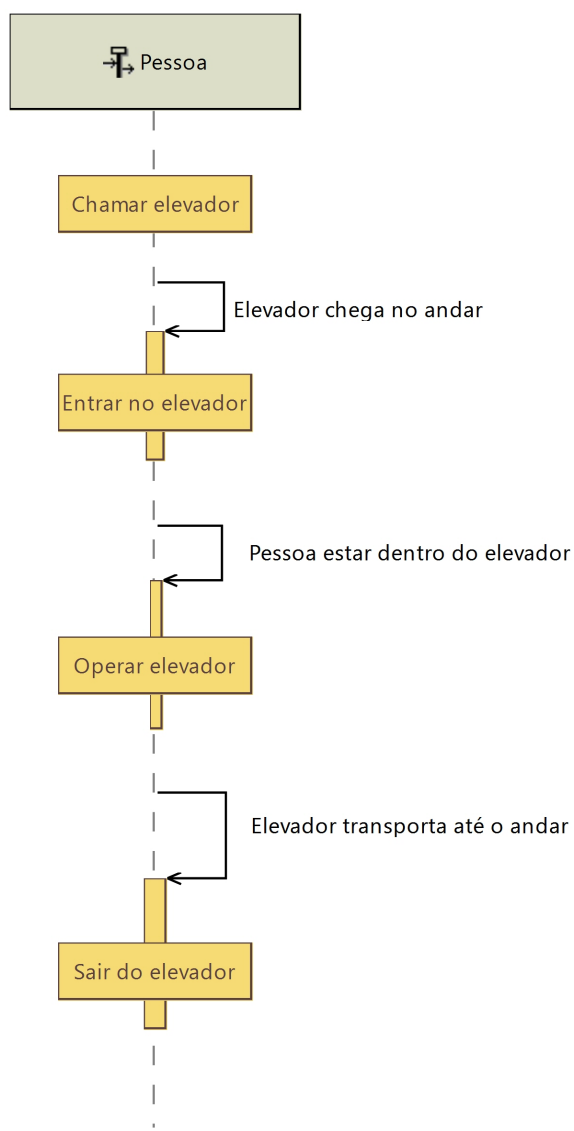


FIGURA 3.3 – Diagrama operacional do elevador

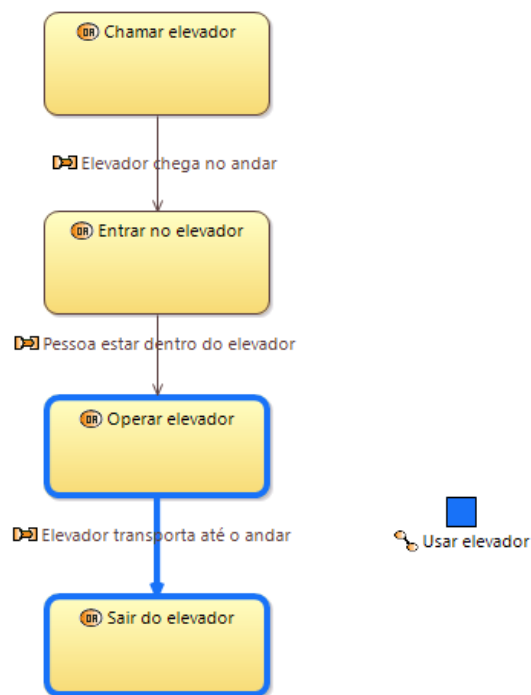


FIGURA 3.4 – Diagrama de descrição das atividades operacionais

Outro diagrama decorrente desse é o de arquitetura operacional, no qual atribuímos a cada ator as atividades operacionais estabelecidas. Nesse caso, o diagrama mostrado na Figura 3.6 representa o ator ativo do sistema, ou seja, aquele que irá interagir com o elevador, e os blocos de atividades contidos no bloco desse ator.

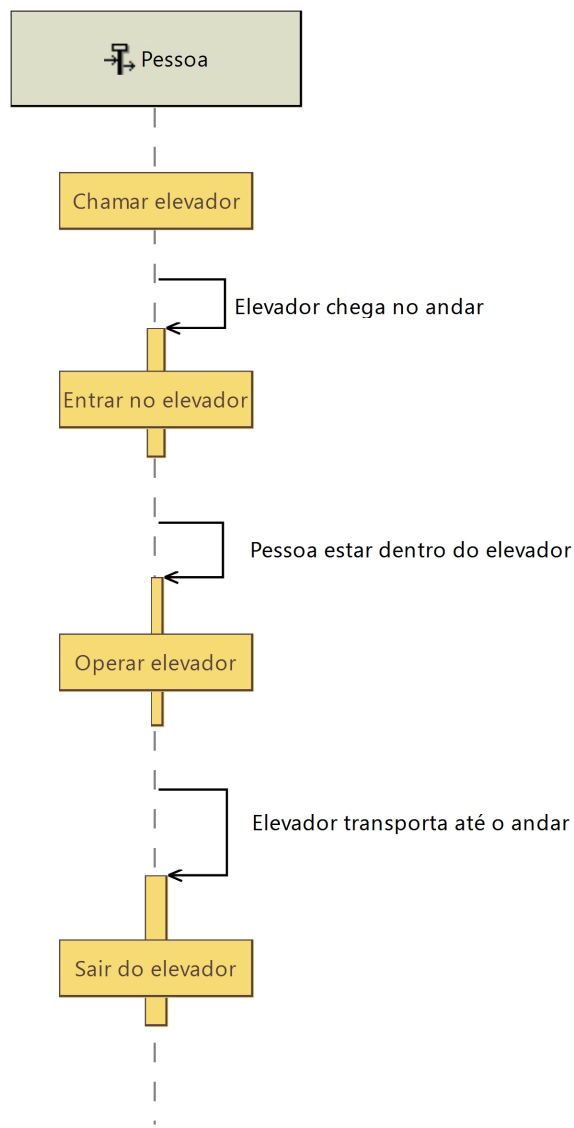


FIGURA 3.5 – Diagrama operacional do elevador

De maneira simples, a pessoa irá chamar o elevador, quando esse chegar no andar, ela irá entrar e operar o sistema, ou seja, designar seu andar destino. Por fim, ela irá sair do elevador quando esse terminar o transporte.

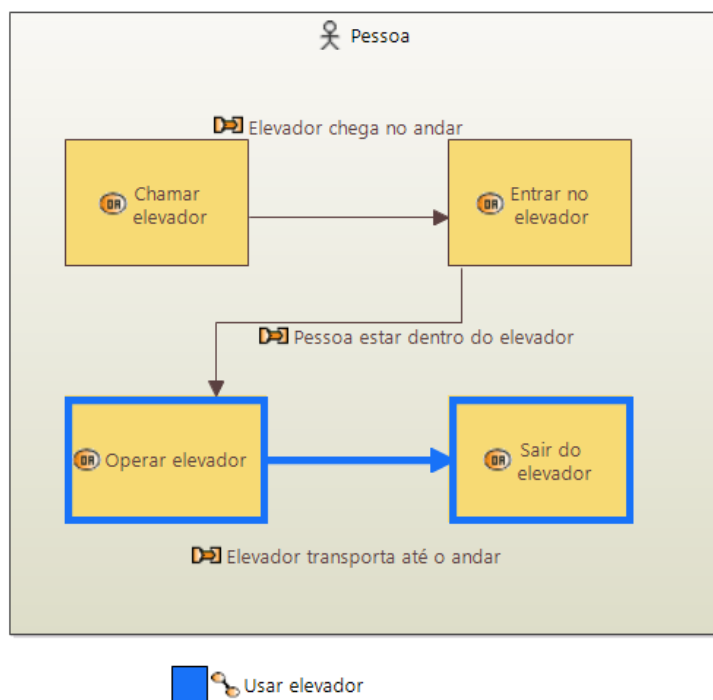


FIGURA 3.6 – Diagrama de arquitetura operacional

Dessa forma, esse diagrama representa a análise de quem são os atores e suas interações com o sistema.

### 3.1.2 Análise do Sistema

O próximo passo foi realizar a análise do sistema. Isso significa começar a identificar os requisitos para se cumprir as missões determinadas e criar fluxos de atividades relações entre atores e sistemas pensando na arquitetura do sistema. Primeiramente, foi necessário inserir o diagrama de contexto do sistema, conforme Figura ???. Para esse caso, definimos o transporte como uma generalização de elevador, e os passageiros como uma generalização de pessoas. Esse passo poderia ser descartado, visto que para esse modelo não há grandes modificações por conta da introdução desse conceito. Mais uma vez, como o objetivo é testar a ferramenta de análise, foi decidido manter esses exemplos dessa forma.

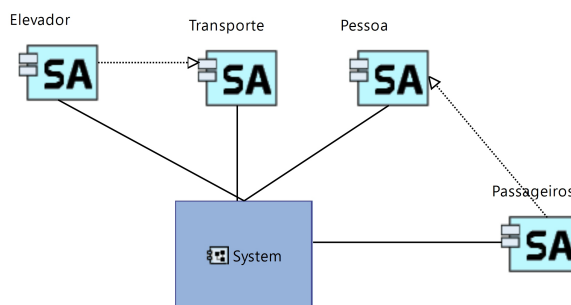


FIGURA 3.7 – Diagrama de contexto do sistema

Foi desenvolvido o diagrama de missão. O elevador tem a missão de transportar pessoas. Ademais, foi acrescentado os demais passageiros no sistema. Além disso, criou-se uma relação de generalização entre Passageiros e Pessoa, visto que, a pessoa também será um passageiro. Para esse caso, é preciso reforçar que a 'Pessoa' é o ponto de vista de quem está operando o elevador, enquanto os demais passageiros são aqueles que estão sendo transportados juntos com ela.

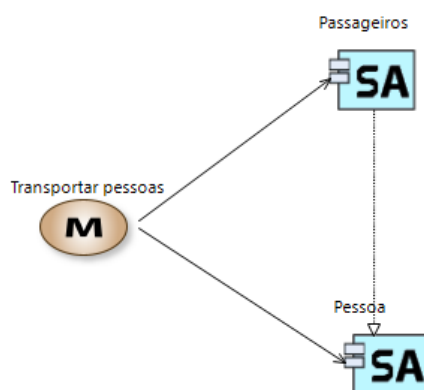


FIGURA 3.8 – Diagrama de Missões

A missão deve ser cumprida por meio da satisfação das capacidades do sistema. Para esse caso, vamos considerar de maneira simples quatro capacidades que devem ocorrer: abrir e fechar as portas para as pessoas entrarem e saírem do elevador, subir e descer, visto que o elevador deve se mover, prover conforto, para as pessoas utilizarem o elevador, e, por fim e não menos importante, prover segurança.

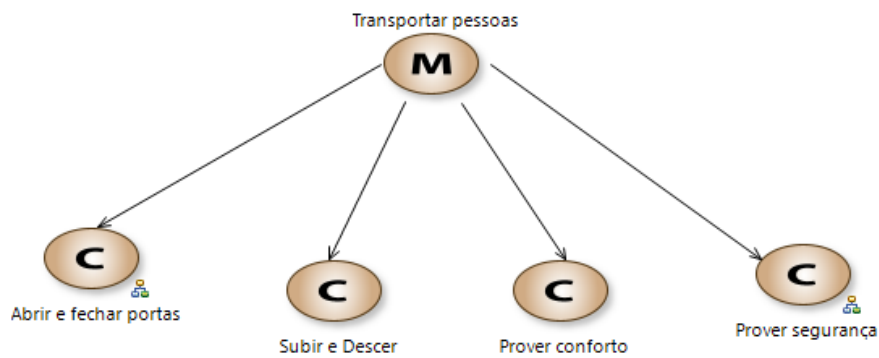


FIGURA 3.9 – Diagrama de Capacidades

Em seguida, identificando as capacidades necessárias para realizar a missão, foram escolhidos alguns cenários de exemplo para desenvolvimento da arquitetura do sistema nessa situação. O primeiro cenário foi pensado no objetivo mais simples do sistema: o elevador está transportando os passageiros. A partir dessa etapa, são selecionadas as funções que cada elemento do sistema irá ter, além dos gatilhos para desencadear as demais ações, conforme mostrado nas Figuras 3.10 e 3.11.

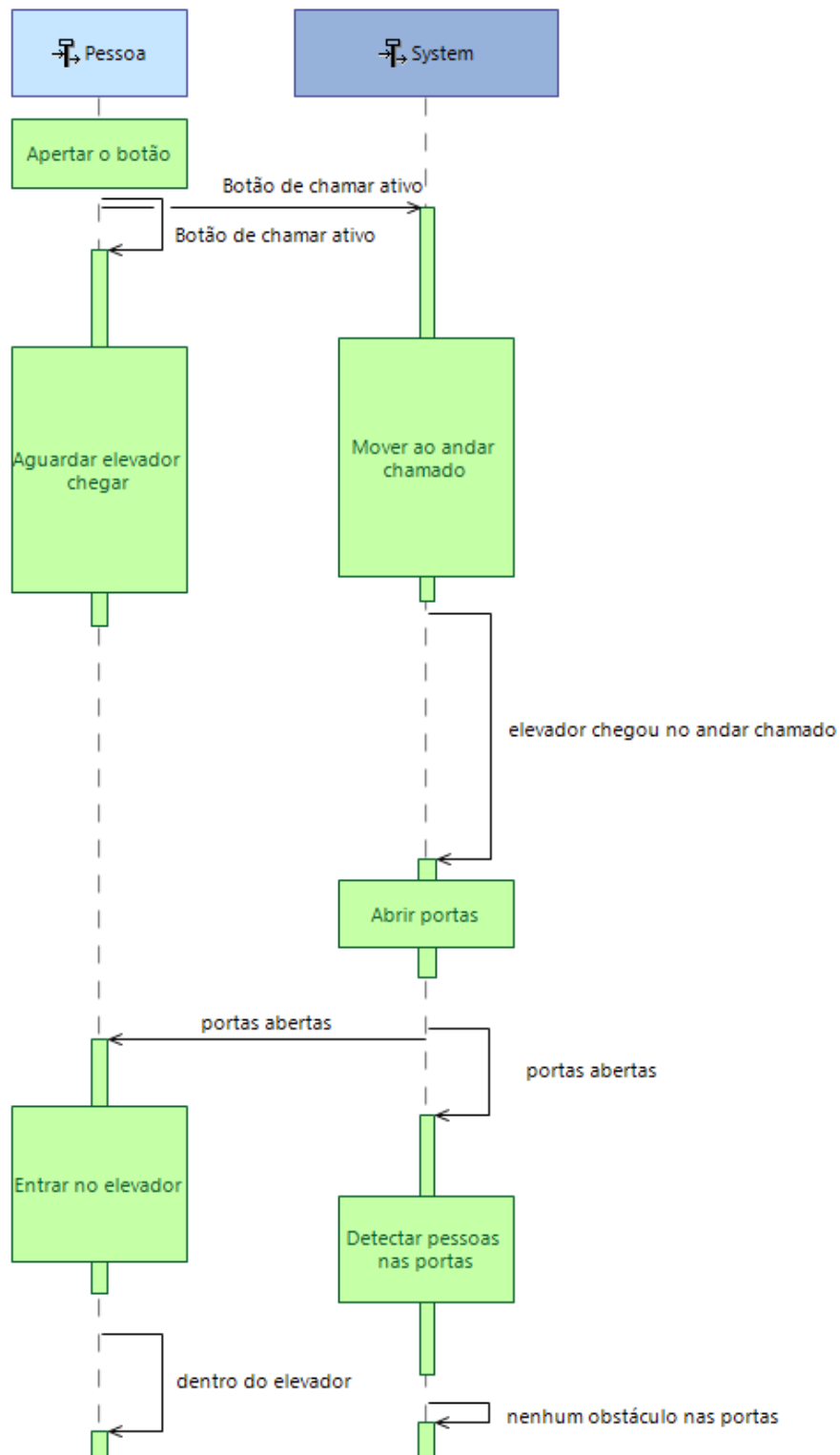


FIGURA 3.10 – Diagrama de intercâmbio do elevador se movendo - parte A



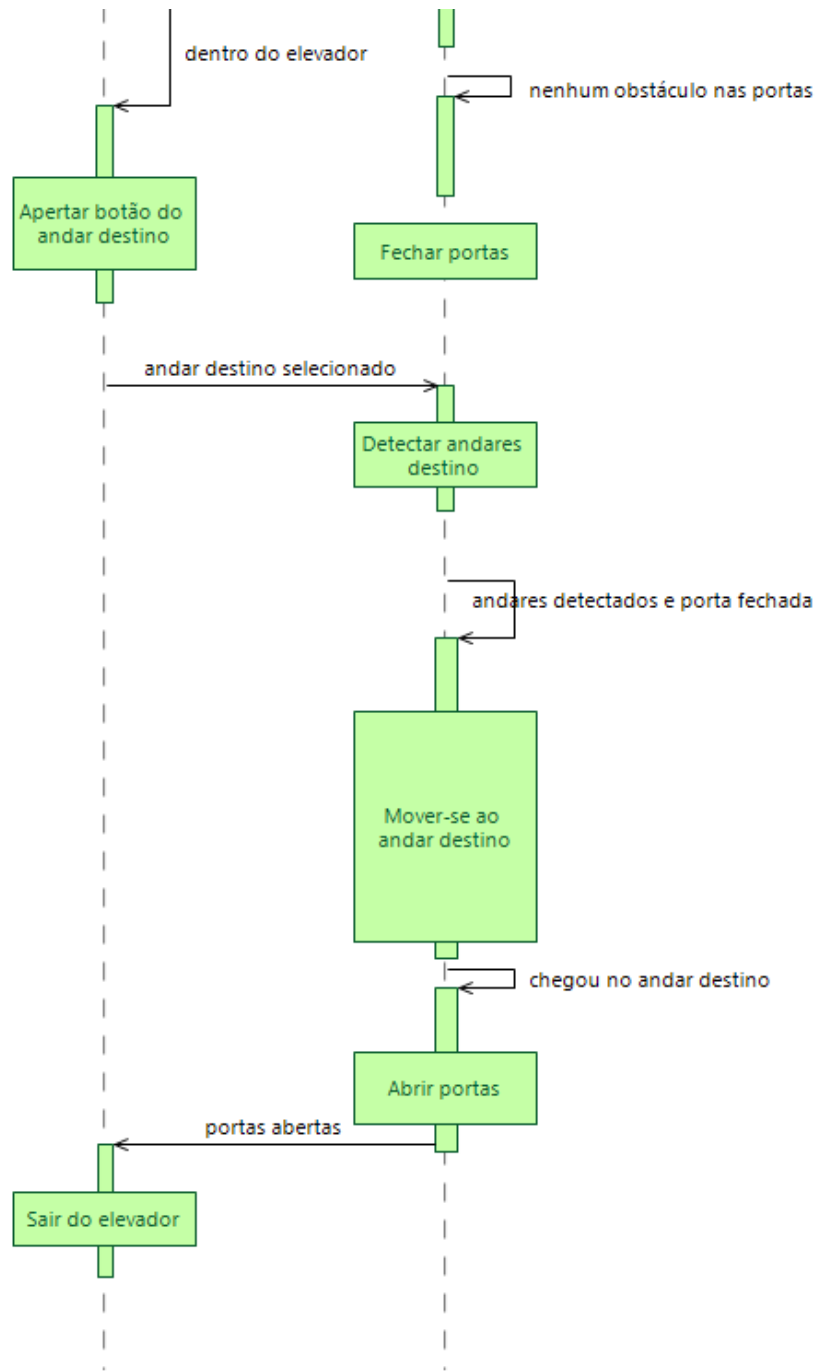


FIGURA 3.11 – Diagrama de intercâmbio do elevador se movendo  
- parte B

Dessa forma, temos o diagrama geral de relações entre as funções arquitetadas para os elementos do sistema, conforme mostrado na Figura 3.12

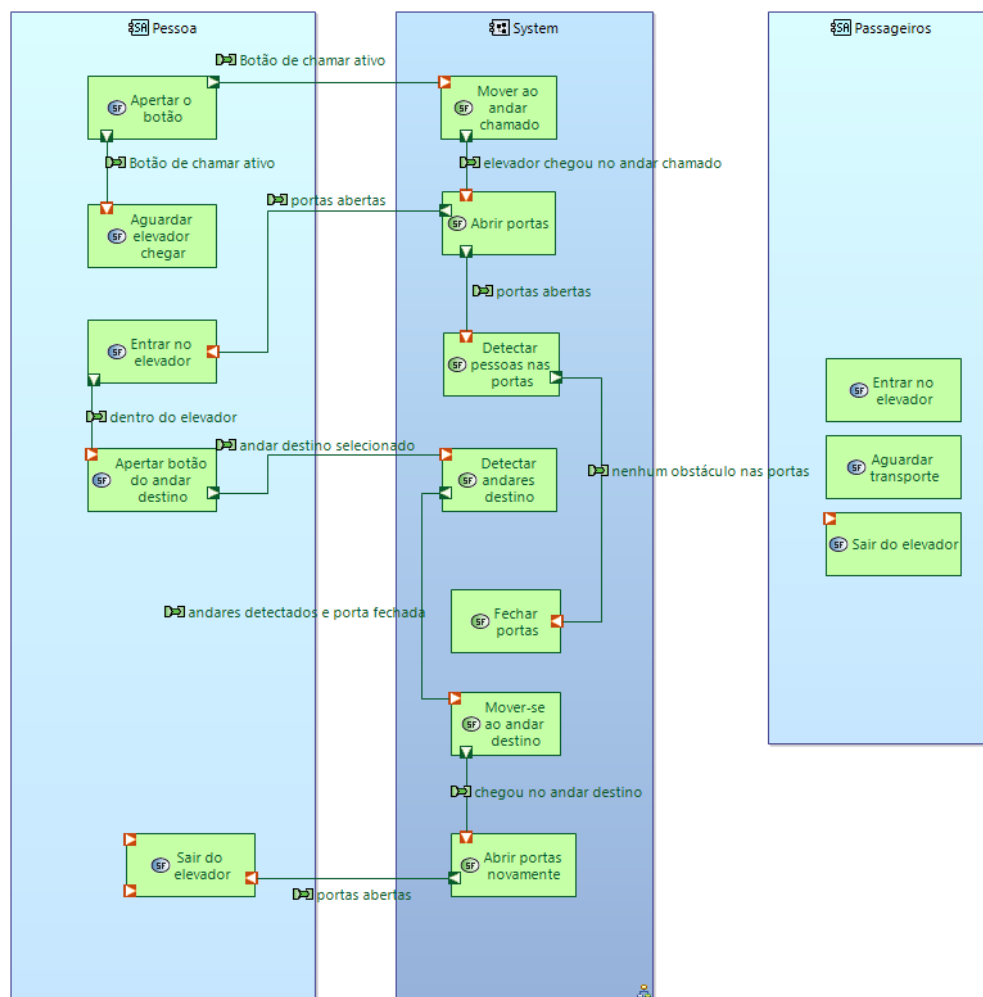


FIGURA 3.12 – Diagrama de arquitetura do elevador se movendo

Para essa situação, os demais passageiros não são de interesse da análise do sistema. Por isso, iremos considerar somente as relações entre funções da 'pessoa' e do sistema. Ao precisar utilizar o elevador, iremos considerar que o primeiro passo é a pessoa apertar o botão do elevador. Em seguida a pessoa irá esperar o deslocamento do elevador para seu andar. Percebe-se que essa ação desencadeia o movimento do elevador para o andar chamado. Caso o elevador estivesse já no andar no qual foi chamado, ele passaria direto para ação de abrir as portas. Todavia, isso não foi representado de maneira direta para simplificar o modelo de estudo.

Os demais passos e relações de funções são apresentados nesse diagrama e consiste, de maneira geral, no elevador transportar a pessoa até o andar destino. Ademais, foi inserido funções para identificar se existe algum obstáculo nas portas. Essa situação foi pensada para evitar que as portas fechassem em cima de algum dos passageiros. A importância do uso de MBSE é justamente para casos como esse: identificamos um problema possível ao longo do ciclo de desenvolvimento do projeto e adicionamos as modificações necessárias

para contornar o problema. Essas modificações afetam o sistema inteiro, e podemos entender como se dará as consequências nas relações das funções presentes nessa arquitetura. O projeto de Engenharia de Sistemas é iterativo. Sendo assim, a cada passo é possível aprimorar e refinar o modelo, tornando mais próximo para resolver todos os problemas encontrados e, por fim, simulando a eficácia da arquitetura proposta.

Considerando o funcionamento das portas e de movimentação do elevador, temos o seguinte diagrama de funções, mostrado na Figura 3.13, em que é possível identificar a corrente de ações para o funcionamento das portas e para movimentação do elevador.

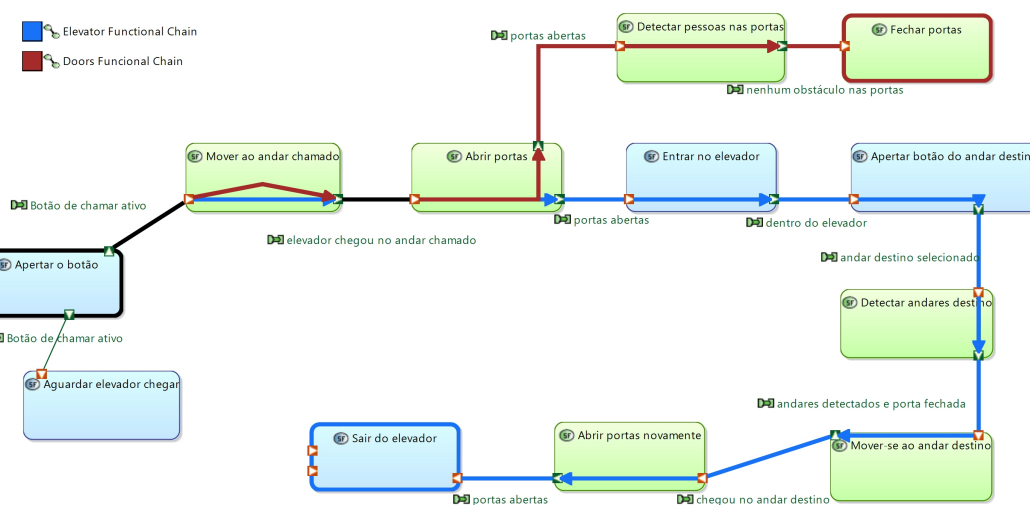


FIGURA 3.13 – Diagrama de funcionamento e relações das funções do sistema

Quando pensamos em projetos mais complexos que o elevador, como é o caso de alguns satélites, a capacidade do aprimoramento iterativo torna-se ainda mais importante, visto que a construção de protótipos de teste aumenta o orçamento do projeto, e o tempo gasto para desenvolvimento dele. Sendo assim, o uso do MBSE possibilita realizar testes no campo teórico, ao propormos arquiteturas pensadas em modelos, justamente como ocorreu no exemplo do elevador. Em um projeto real, seriam necessários mais iterações de aprimoramento, identificando todos os requisitos do sistema que ainda não foram apresentados e a arquitetura para satisfazer esses requisitos. Todavia, esse processo é longo e necessita de conhecimentos técnicos mais específicos sobre elevadores, fugindo do escopo desse trabalho. Portanto, manteremos esse modelo o mais simplificado possível.

Em seguida, iremos utilizar uma segunda modelagem para atribuir uma arquitetura para cumprir mais uma capacidade necessária ao sistema: prover segurança. Vamos supor um cenário em que existe uma maior quantidade de pessoas que o motor do elevador suporta para transportar de maneira segura. Para evitar um acidente, é necessário que o sistema do elevador identifique essa possível sobrecarga para travar o elevador e avisar aos passageiros da situação. Sendo assim, conforme mostrado na Figura 3.15, temos a

situação hipotética de mais de seis passageiros na cabine. Cada elevador tem capacidade de carga diferentes dependendo do uso. Sabe-se que alguns suportam mais de 20 pessoas, e são utilizados em prédios comerciais. Todavia, tais elevadores são muito mais caros. Em uma situação mais provável, vamos considerar elevadores de uso residencial mais simples, que suportam no máximo 6 passageiros. Ademais, vamos utilizar como parâmetro a quantidade de passageiros, mas seria facilmente modificável para peso em quilogramas, caso necessite de um modelo mais técnico,

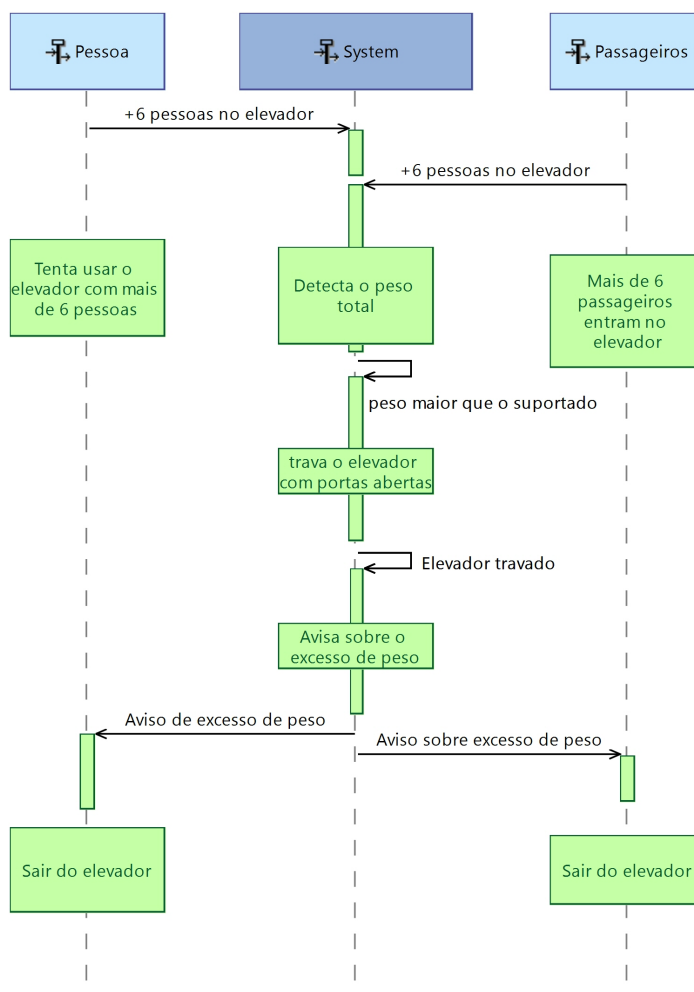


FIGURA 3.14 – Cenário em que o elevador está com sobrepeso

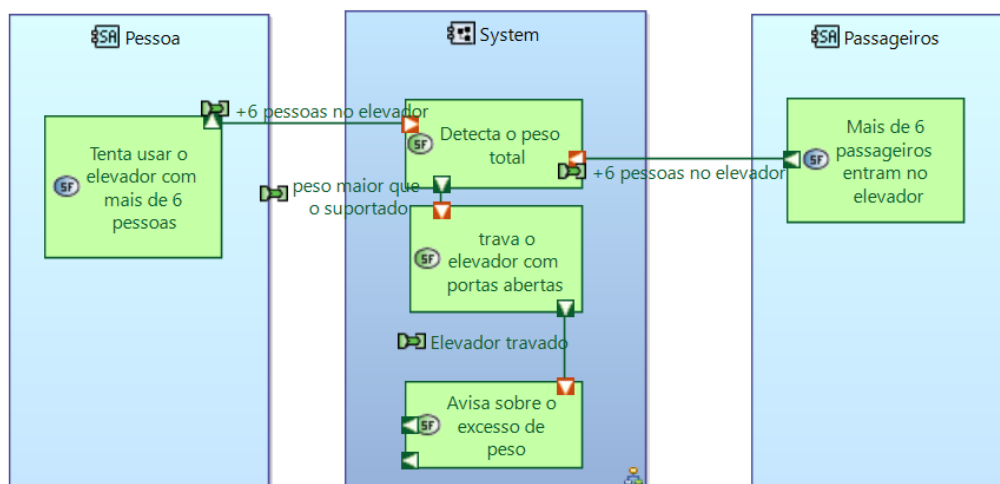


FIGURA 3.15 – Diagrama de arquitetura do elevador em sobre-carga

Conforme mostrado no diagrama de arquitetura para essa situação, o elevador irá travar com as portas abertas em caso de sobrepeso, para permitir que o excesso de passageiros saia de maneira segura da cabine.

### 3.1.3 Arquitetura Lógica

Por fim, foram obtidos os diagramas da análise lógica do sistema. Primeiramente, obteve-se o diagrama dos elementos existentes, conforme mostrado no Figura 3.16

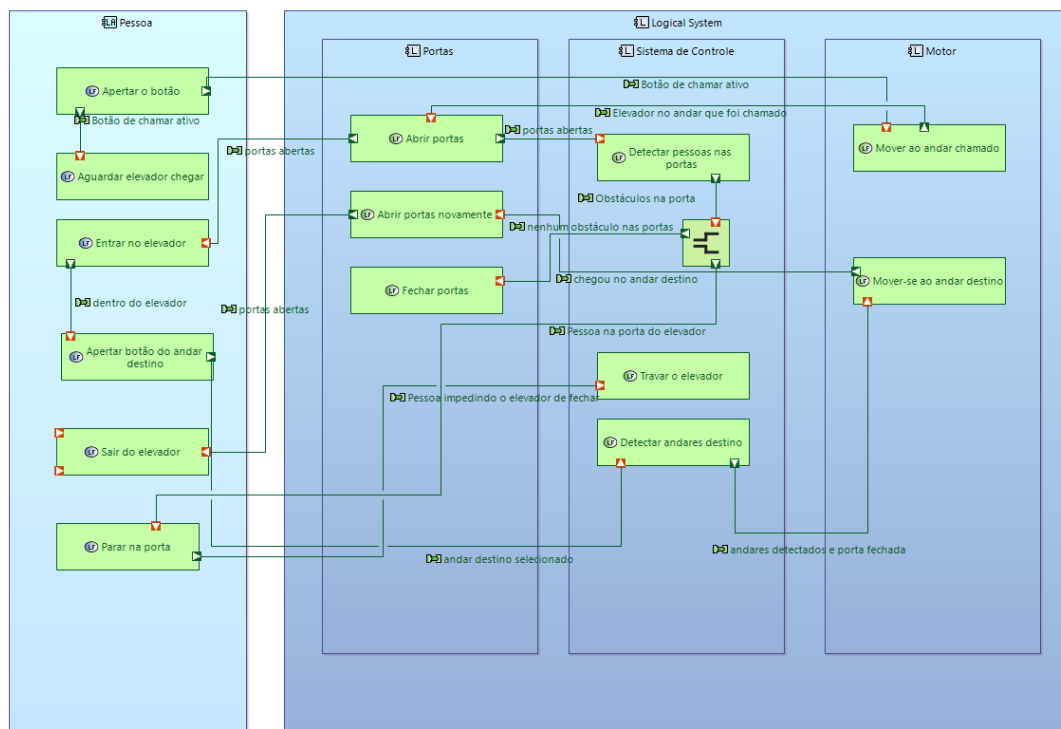


FIGURA 3.16 – Arquitetura Lógica

Em seguida, para ter uma visualização mais macro do sistema, agrupou-se as funções lógicas conforme mostrado na Figura 3.17.

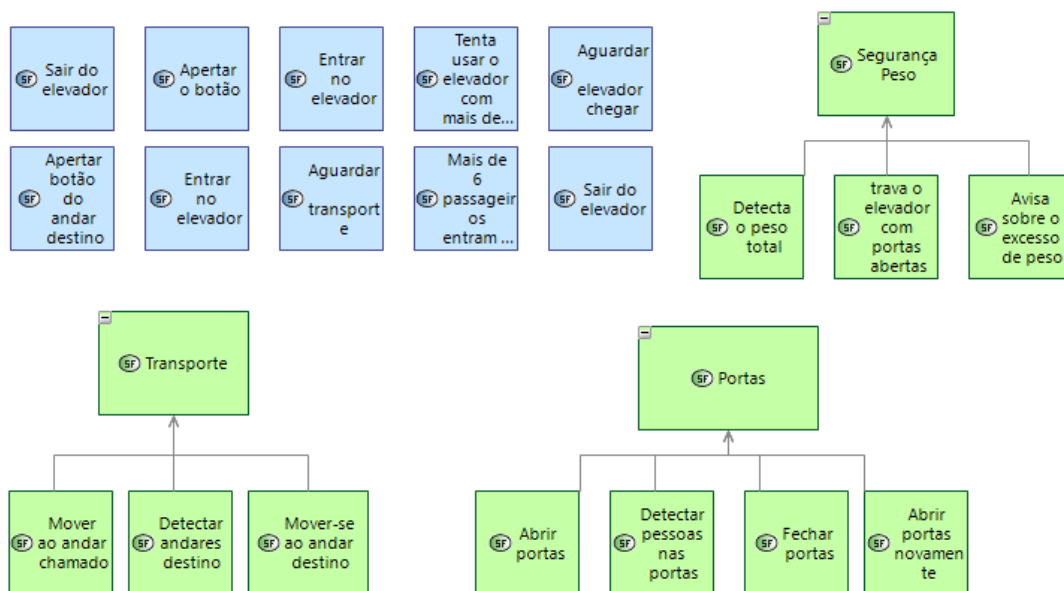


FIGURA 3.17 – Composição das funções do sistema

Dessa forma, é possível obter o diagrama da arquitetura lógica do sistema, mas utilizando as funções macro criadas a partir do agrupamento, conforme mostrado na Figura

3.16. Sendo assim, temos o sistema lógico do elevador.

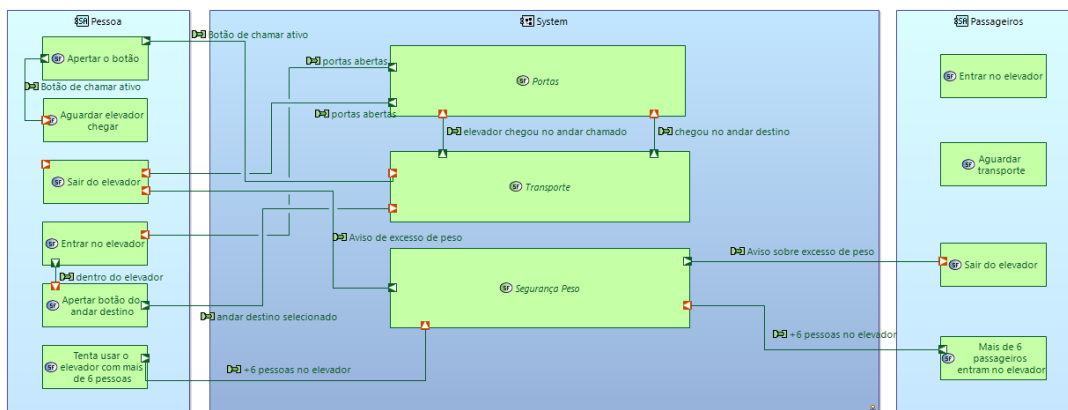


FIGURA 3.18 – Sistema Lógico do Elevador

Por fim, foi possível utilizar o Capella para criar um modelo de máquina de estados para o sistema, utilizando como gatilhos as funções lógicas criadas anteriormente, conforme mostrada na Figura 3.19

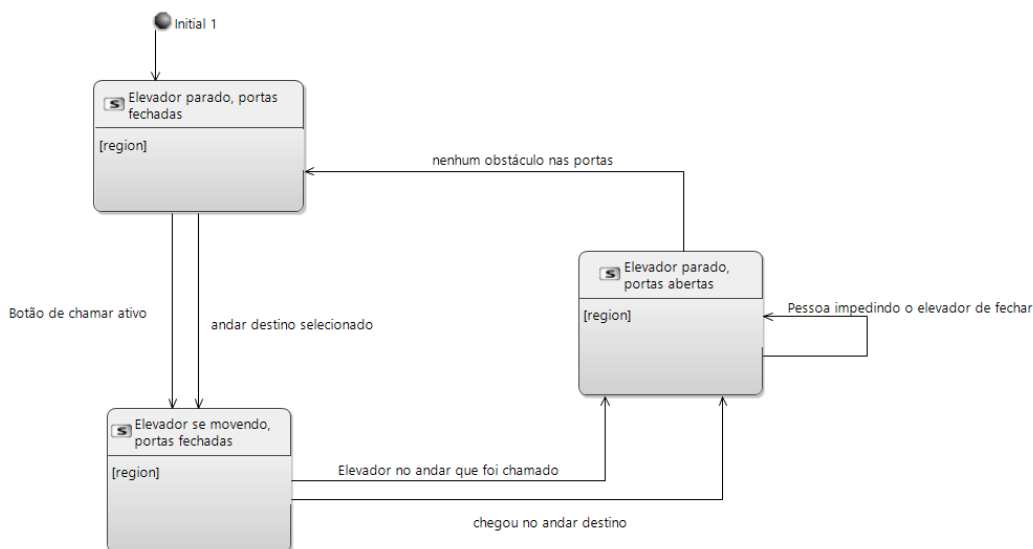


FIGURA 3.19 – Máquina de Estados do sistema de movimentação do elevador

Utilizamos, de maneira simplificada, somente três estados para o elevador. O modelo utilizado considera somente o elevador se movendo como um único estado. Todavia, um próximo passo seria refinar o modelo para identificar nos diagramas a diferença de quando o elevador está subindo ou descendo. Para isso, seria necessário utilizar um comparador lógico para verificar se os andares selecionados são maiores ou menores do que o andar atual.

Também seria necessário, do ponto de vista prático, desenvolver a lógica do elevador para quando esse está sendo atribuído mais de um andar de destino. Por exemplo, se a cabine estiver no terceiro andar, mas dentro estiver selecionado o térreo e o andar cinco, qual movimento o elevador deverá fazer? O refinamento das máquinas de estado permite desenvolver esse tipo de lógica. Essa é uma das principais vantagens do uso desse método: validar a arquitetura atual e propor modificações para suprir os requisitos mais detalhados.

## 3.2 Utilização da biblioteca `sismic` para aplicação de MBSE

Outro método de aplicação de MBSE utilizado foi a criação de Máquinas de Estado utilizando a biblioteca em linguagem Python `sismic`. Para isso, primeiramente passou-se o modelo de máquina de estados desenvolvido para linguagem YAML, conforme mostrado na Figura 3.20.

```
statechart:
  name: Elevador
  preamble: |
    movendo = False
    doors_open = False

  root state:
    name: ativo
    initial: parado_portasfechadas
    states:
      - name: parado_portasfechadas
        transitions:
          - target: movendo_portasfechadas
            guard: botão_chamar_ativo
            action: |
              movendo = true
              doors_open = False
          - target: movendo_portasfechadas
            guard: andar_destino_selecionado
            action: |
              movendo = true
              doors_open = False
      - name: movendo_portasfechadas
        transitions:
          - target: parado_portasabertas
            guard: elevador_no_andar_chamado
            action: |
              movendo = False
              doors_open = True
          - target: parado_portasabertas
            guard: chegou_ao_destino
            action: |
              movendo = False
              doors_open = True
      - name: parado_portasabertas
        transitions:
          - target: parado_portasabertas
            guard: pessoa_impedindo_porta
            action: doors_open = True
          - target: parado_portasfechadas
            guard: nenhum_obstaculo_nas_portas
            action: doors_open = False
```

FIGURA 3.20 – Código em YAML utilizado para definição da máquina de estado para sistema elevador



A linguagem possibilita, de maneira fácil, criar uma máquina de estados, atribuindo as transições e os parâmetros para cada estado. Por exemplo, conforme visto anteriormente, utilizamos um modelo de somente três estados para facilitar a comparação do estudo. É possível atribuir o nome para cada um desses estados. As transições possíveis para cada estado é definido dentro de 'transitions', onde primeiramente defini-se o alvo, 'target', ou seja, o estado seguinte. Além disso, define-se as condições para ocorrer essas transições. No código, essa definição é feita utilizando o argumento "guard". Por fim, temos as ações definidas em 'action'. Para esse exemplo, as mudanças de estados estão alterando somente dois parâmetros: portas e movimentação do elevador.

Apesar do exemplo simples, essa linguagem permite o desenvolvimento de Máquina de Estados para sistemas muito mais complexos. Ela permite também inserir de maneira rápida sub-estados para cada estado já desenvolvido, utilizando para isso a hierarquização dos estados. Também é possível desenvolver estados em paralelo. Todavia, o uso dessa linguagem para criação do modelo não é tão eficiente quanto o uso do software Capella, devido a dificuldade de visualização do modelo em tempo real. O Capella permite a introdução direta dos símbolos de estados e transições da máquina de estados, utilizando já as funções e gatilhos implementadas nas arquiteturas lógicas e análise operacional do sistema. O uso da biblioteca sismic tem vantagem não na criação do modelo do zero, mas sim na validação desse modelo por meio de simulações em python, por exemplo.

Ainda sim, foi possível obter o diagrama em linguagem SysML a partir da exportação do diagrama para o software PlantUML, utilizando o código em python mostrado na Figura 3.21.

```
from sismic.io import import_from_yaml
from sismic.io import export_to_plantuml
from sismic.model import Statechart

with open('elevador2.yaml') as f:
    statechart = import_from_yaml(f)
    assert isinstance(statechart, Statechart)

plantuml = export_to_plantuml(statechart, filepath='')

arquivo = open('plantuml_1.txt', 'w')
arquivo.write(plantuml)
```

FIGURA 3.21 – Código em python utilizando a biblioteca sismic para exportação do código PlantUML

O código exportado foi salvo em arquivo 'txt' e utilizado no site do software PlantUML, mostrado na Figura 3.22.

```

@startuml
title Elevador
state "ativo" as ativo {
[*] --> paradoportasfechadas
state "parado_portasabertas" as paradoportasabertas {
paradoportasabertas --> paradoportasabertas : [pessoa_impedindo_porta] / doors_open = True
paradoportasabertas --> paradoportasfechadas : [nenhum_obstaculo_nas_portas] / doors_open = False
}
state "movendo_portasfechadas" as movendoportasfechadas {
movendoportasfechadas --> paradoportasabertas : [elevador_no_andar_chamado] / movendo = False; doors_open = True
movendoportasfechadas --> paradoportasfechadas : [chegou_ao_destino] / movendo = False; doors_open = True
}
state "parado_portasfechadas" as paradoportasfechadas {
paradoportasfechadas --> movendoportasfechadas : [botao_chamar_ativo] / movendo = true; doors_open = False
paradoportasfechadas --> movendoportasfechadas : [andar_destino_selecionado] / movendo = true; doors_open = False
}
}
@enduml

```

[//www.plantuml.com/plantuml/png/dP91OyCm38N1\\_XMYnq7PVMZb-qNx5QAKMDj3TXMiEHUHI\\_xrbWvX8HHUahmm-yzVV9U581M59ts6WdU8rqmS76Yg0GRFx26q21](http://www.plantuml.com/plantuml/png/dP91OyCm38N1_XMYnq7PVMZb-qNx5QAKMDj3TXMiEHUHI_xrbWvX8HHUahmm-yzVV9U581M59ts6WdU8rqmS76Yg0GRFx26q21)  
 Submit Switch layout View as PNG View as SVG View as ASCII Art

FIGURA 3.22 – Código em PlantUML utilizado para visualização da máquina de estado para sistema elevador

Por fim, utilizando essas ferramentas, foi possível não só definir a Máquina de Estados em python, como visualizar o resultado no diagrama SysML, conforme Figura 3.23.

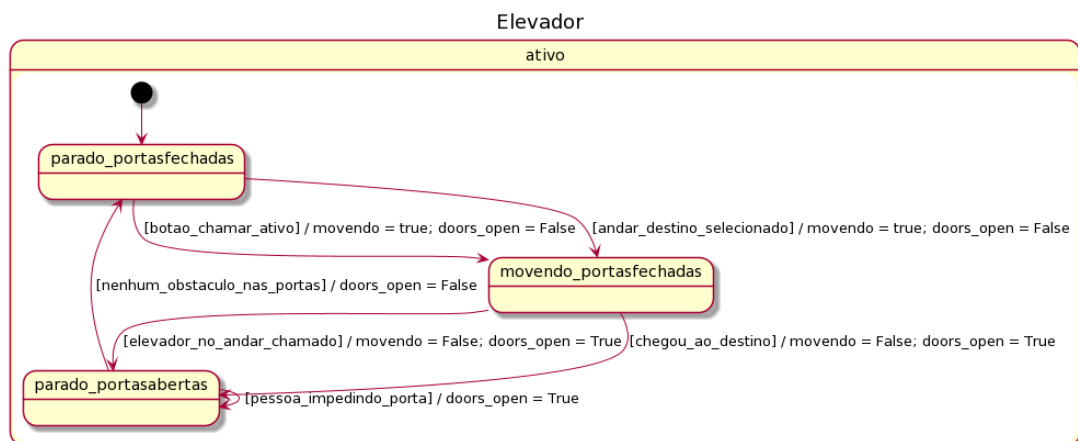


FIGURA 3.23 – Máquina de Estados para Elevador em linguagem SysML

Dessa forma, a partir de um exemplo simples do dia a dia, que é o caso do elevador, foi possível verificar a eficácia do uso de metodologia MBSE para engenharia de sistemas, possibilitando a organização e otimização do desenvolvimento de projetos de diversas áreas.

## 4 Conclusão

A Engenharia Aeroespacial mostra-se desafiadora do ponto de vista de complexidade de sistemas. Mesmo projetos simples, como 'cubo-sats' apresentam diversos subsistemas integrados, como controle de atitude, controle de bordo, eletrônica e potência, estrutura, entre outros. Todos esses sistemas estão conectados por fluxos físicos, como correntes elétricas em um satélite ou fluido oxidante num motor foguete, como também por fluxo de informações, em controladores e antenas. A quantidade de interfaces entre esses subsistemas aumenta o risco de problemas que poderiam ocorrer ao longo do desenvolvimento do projeto. Todavia, a Engenharia de Sistemas se prova um método eficiente para contornar esses riscos. O MBSE, baseado em modelos, mostra-se uma solução prática para validar modelos de arquiteturas e tentar testar soluções sem a necessidade de desenvolver um protótipo físico. A possibilidade de realizar melhorias ao longo da aplicação iterativa do método permite antecipar e testar soluções alternativas, melhorando o projeto. Isso foi observado no exemplo do elevador, em que foi possível realizar algumas melhorias já, mas prever possíveis detalhamentos de arquiteturas que poderiam ser desenvolvidas num projeto real. Por fim, vimos que existem soluções como o software Capella e a biblioteca Sismic que possibilitam a definição de máquinas de estado para identificação de um modelo válido de testes. Esses dois softwares apresentam vantagens e desvantagens, mas juntos se complementam para satisfazer as necessidades da aplicação da metodologia MBSE. O Capella permite o desenvolvimento de um projeto do zero de maneira intuitiva e organizada, apesar de necessitar de uma curva de aprendizagem para uso eficiente de todas as múltiplas ferramentas disponíveis. A separação de uma visão operacional, focando em satisfazer os requisitos do clientes, de uma visão de solução, pensando na arquitetura do projeto permite evitar viés de engenharia e facilitar a implementação de novas soluções e melhorias de projeto ao longo do ciclo de vida deste. Além disso, ele possui simplificações eficientes que facilitam o uso de MBSE em comparação a linguagem pura em SysML. Por outro lado, o Sismic aparenta uma solução melhor pensando em validação de projeto, visto que o Python é uma linguagem mais flexível, que permite o uso integrado com diversas outras bibliotecas e outros softwares. Além disso, como Python apresenta uma linguagem mais intuitiva e comumente ensinada em muitas engenharias, a curva de aprendizagem pode se tornar mais simples para alguns engenheiros. Por não utilizar uma metodologia

---

clara como o Arcadia, tem-se uma maior flexibilidade e diversificação de soluções possíveis, principalmente pensando em modelos de mais baixo nível, e com mais detalhes. Sendo assim, é evidente o potencial do uso do MBSE no desenvolvimento de projetos no geral, sendo ainda mais interessante a aplicação em sistemas complexos, como é o caso da maioria dos projetos do setor aeroespacial.

# Referências

CAPELLA. Disponível em: <<https://www.eclipse.org/capella/arcadia.html>>. Acesso em: 06 jun. 2020.

ENGENHARIA Aeroespacia. Disponível em: <<http://www.ita.br/aer/aesp>>. Acesso em: 04 jun. 2020.

EQUIVALENCES and differences between Arcadia/Capella and SysML. Disponível em: <<https://www.youtube.com/watch?v=qFfj8K3uj3Y>>. Acesso em: 16 out. 2021.

HAREL, D. Statecharts: A visual formalism for complex systems. **Science of Computer Programming** **8**, p. 231–274, 1987.

HOLT, S. P. J. **SysML for Systems Engineering**. 3ed. ed. Londres: The Institution of Engineering and Technology, 2018.

MITRE. **MITRE Systems Engineering Guide**. 1ed. ed. [S.l.]: MITRE Corporate Communications and Public Affairs, 2014.

MOORE, R. S. A. **A Practical Guide to SysML**. 3ed. ed. [S.l.]: Elsevier, 2011.

RODRIGUES, L. E. **Fundamentos da engenharia aeronáutica**. 1ed. ed. São Paulo: Cengage Learning, 2013.

SISMIC user manual. Disponível em: <<https://sismic.readthedocs.io/en/latest/index.html>>. Acesso em: 31 out. 2021.

WERTZ DAVID F. EVERETT, J. J. P. J. R. **Space Mission Engineering: The New Smad**. 1ed. ed. [S.l.]: CMicrocosm Press, 2011.

YAML Ain't Markup Language (YAML™) version 1.2. Disponível em: <<https://yaml.org/spec/1.2.2/>>. Acesso em: 31 out. 2021.

## FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO <p style="text-align: center;">TC</p>	2. DATA 26 de novembro de 2021	3. REGISTRO N° DCTA/ITA/TC-135/2021	4. N° DE PÁGINAS 51
5. TÍTULO E SUBTÍTULO: Transformação de modelos sistêmicos para simulação discreta			
6. AUTOR(ES): <b>Lucas Lenzi Alves</b>			
7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES): Instituto Tecnológico de Aeronáutica - ITA			
8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR: Sistemas; Capella; Máquina; Estados			
9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: Sistemas aeroespaciais; Projetos; Engenharia de sistemas; Engenharia aeroespacial			
10. APRESENTAÇÃO: <span style="float: right;">( X ) Nacional ( ) Internacional</span> ITA, São José dos Campos. Curso de Graduação em Engenharia Aeroespacial. Orientador: Prof. Dr. Christopher S. Cerqueira. Publicado em 2021.			
11. RESUMO: Engenharia Aeroespacial consiste na concepção de sistemas aeroespaciais como satélites, foguetes, espaçonaves, entre outros. São sistemas que integram tecnologias complexas, e necessitam, em sua maioria, de métodos para otimizar os projetos desenvolvidos, com objetivo de minimizar riscos, economizar financeiramente e reduzir o tempo de desenvolvimento. Para isso, a Engenharia de Sistemas torna-se extremamente útil ao ser utilizada em projetos aeroespaciais. Esse campo interdisciplinar de engenharia permite um foco nessa otimização de desenvolvimentos de projetos, podendo atuar em toda a linha do tempo. De maneira resumida, essa engenharia busca relacionar os requisitos do usuário com modelos de soluções possíveis de arquiteturas, realizando processos iterativos para melhorar a solução final. Com o passar dos anos e, em consequência dos avanços tecnológicos, uma metodologia vem sendo utilizada com relevância em projetos complexos: Engenharia de Sistemas Baseada em Modelos (MBSE). O objetivo desse trabalho é de compreender a eficácia e vantagens do uso dessa metodologia no dia a dia do engenheiro de sistemas, e comparar com possíveis usos no setor aeroespacial. A partir da construção de modelos para sistemas do dia a dia, como um elevador, foi possível projetar um sistema básico e entender seu comportamento durante o funcionamento. Foi utilizado software Capella, que utiliza a metodologia Arcadia como solução para aplicação do método MBSE. Além disso, comparou-se com o uso de outros softwares e linguagens de engenharia de sistema, como os diagramas puros em SysML.			
12. GRAU DE SIGILO: <p style="text-align: center;">( X ) OSTENSIVO ( ) RESERVADO ( ) SECRETO</p>			