UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VANDERSON HAFEMANN FRAGAL

Engenharia de aplicação para sistemas embarcados: transformando especificações SysML em Simulink

Maringá
2013

VANDERSON HAFEMANN FRAGAL

Engenharia de aplicação para sistemas embarcados: transformando especificações SysML em Simulink

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dra. Itana Maria de Souza Gimenes
Co-orientador: Prof. Dr. Edson Alves de Oliveira Júnior

Maringá
2013

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

F811e

Fragal, Vanderson Hafemann
Engenharia de aplicação para sistemas embarcados : transformando especificações SysML em Simulink / Vanderson Hafemann Fragal. -- Maringá, 2013.
104 f. : il. col., figs., tabs.

Orientadora: Prof.ª Dr.ª Itana Maria de Souza Gimenes.
Co-orientador: Prof. Dr. Edson Alves de Oliveira Júnior.
Dissertação (mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2013.

1. Linha de produto de software. 2. SysML. 3. Simulink. 4. Sistemas embarcados. 5. Veículos aéreos não tripulados (VANT). 6. Software - Desenvolvimento dirigido por modelos. I. Gimenes, Itana Maria de Souza, orient. II. Oliveira Júnior, Edson Alves, coorient. III. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

CDD 22.ed.005.45
AMMA-00646

FOLHA DE APROVAÇÃO

VANDERSON HAFEMANN FRAGAL

Engenharia de aplicação para sistemas embarcados: transformando especificações SysML em Simulink

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Profa. Dra. Itana Maria de Souza Gimenes
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Elisa Hatsue Moriya Huzita
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Adenilso da Silva Simão
Universidade de São Paulo – ICMC/USP

Aprovada em: 19 de fevereiro de 2013.
Local da defesa: Sala 102, Bloco C56, *campus* da Universidade Estadual de Maringá

AGRADECIMENTO(S)

Agradeço a Deus, por me dar a vida, saúde, motivação e força não apenas para que este trabalho pudesse ser realizado, mas também para que o mesmo fosse realizado com a maior satisfação possível, já que neste tempo, a vida e o mestrado funcionaram em paralelo, e não sequêncialmente.

Agradeço a minha orientadora, Profa. Dra. Itana Maria de Souza Gimenes, por sua sábia orientação, por sua exigência pela perfeição e dedicação, que me fizeram crescer enormemente. Tenho por ela uma grande admiração, seja por sua imensa capacidade técnica, quanto por sua integridade e honestidade como profissional e cidadã. Também agradeço ao professor Edson Alves e Oliveira Júnior pela atenção dada como coorientador desse trabalho.

Agradeço aos membros do comitê avaliador deste trabalho, a Profa. Dra. Elisa Hatsue Moriya Huzita, o Prof. Dr. Adenilso da Silva Simão e o Prof. Dr. Sergio Roberto Pereira da Silva pelo tempo e empenho, pois não só avaliaram, mas também auxiliaram no enriquecimento deste trabalho com comentários e discussões de importante valor.

A todos os professores do Departamento de Informática da UEM, á Maria Inês e o Wagner que direta ou indiretamente contribuíram com a minha formação e com este trabalho.

Agradeço aos amigos de mestrado Rogerio Ferreira da Silva, Maycon Luís Capellari e Euclides Alfredo Matusse. Agradeço-lhes pelos diversos tipos de apoio oferecido, de contribuições técnicas à própria convivência diária.

Aos amigos e colegas do ICMC-USP de São Carlos, por me acolher, principalmente ao Adenilso, Kalinka, Natássya, Rayner e Daniel, pelo conhecimento compartilhado e pela ajuda fundamental no início dos trabalhos feitos no LSEC.

Não poderia deixar de agradecer a meus queridos pais, João Luiz Fragal e Regina Hafemann Fragal, por sempre quererem o melhor para mim e por me apoiarem ao mudar pra tão longe em busca de um ideal, mesmo que eles tenham sofrido por isto. Agradeço aos meus irmãos Vanessa Hafemann Fragal, Elizângela Hafemann Fragal e Everton Hafemann Fragal, por me apoiarem nesta jornada.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e ao projeto PROCAD, pelo apoio financeiro despendido a este trabalho. Enfim, a todas as pessoas que contribuíram de alguma ou outra forma para que eu chegasse até aqui, o meu mais sincero agradecimento.

Engenharia de aplicação para sistemas embarcados: transformando especificações SysML em Simulink

RESUMO

A evolução das plataformas de hardware transferiu uma grande quantidade de funcionalidades ao software de sistemas embarcados, aumentando sua complexidade. Abordagens como Model Driven Engineering (MDE) e Linha de Produto de Software (*Software Product Line - PL*) podem melhorar o desenvolvimento de sistemas embarcados por meio do uso de linguagens de especificação diferentes de acordo com os níveis de abstração, e de formas de gerenciar variabilidades ao longo do desenvolvimento. A abordagem SyMPLES apoia a concepção de PLs baseadas em SysML. SyMPLES inclui duas extensões SysML, criados pelo mecanismo de perfis UML, tanto para expressar os conceitos variabilidade de PL quanto para associar blocos SysML com as principais classes de blocos funcionais. SyMPLES foi criada inicialmente com foco na atividade de engenharia de domínio da PL, pois os modelos gerados na atividade de engenharia de aplicação não são refinados. Esta dissertação apresenta um processo de transformação de modelos SysML para modelos Simulink que visa complementar a abordagem SyMPLES. Os modelos SysML configurados são utilizados para criar blocos funcionais e máquinas de estado para o Simulink e Stateflow respectivamente. Um exemplo de aplicação foi desenvolvido para um subsistema de uma placa controladora do piloto automático utilizado em veículos aéreos não tripulados, chamado Yapa 2 do projeto Paparazzi. Esta placa é utilizada no contexto do Instituto Nacional de Ciência e Tecnologia para Sistemas Embarcados Críticos (INCT-SEC). Os resultados mostram que os modelos SysML configurados podem ser transformados a fim de representar o sistema com blocos funcionais, que facilita a geração de código.

Palavras-chave: Linha de Produto de Software, SysML, Simulink, sistemas embarcados, veículos aéreos não tripulados, transformação de modelos.

Application engineering for embedded systems: transforming SysML specification in Simulink

ABSTRACT

The evolution of hardware platforms has transferred a great amount of functionality to embedded software, thus increasing its complexity. Model Driven Engineering (MDE) and Software Product Line (PL) can enhance the development of embedded systems by means of using different specification languages according to the abstraction levels and and ways to manage variability across development. The SyMPLES approach suports the conception of SysML-based PLs. It includes two SysML extensions, created by means of the UML profiling mechanism both to express PL variability concepts and to associate SysML blocks to the main classes of functional blocks. SyMPLES was initially created with focus on domain engineering activity of PL, since the models generated in the application engineering activity are not refined. This dissertation presents one transformation process from SysML to Simulink models that is intended to supplement the SyMPLES approach. SysML models created in the PL application engineering activity from SyMPLES are used to create functional blocks and state machines. An application example was developed for one subsystem of an autopilot board used in Unmanned Aerial Vehicles, named Yapa 2 of Paparazzi project, which was studied into the context of National Institute of Science and Technology for Safety Critical Embedded Systems (INCT-SEC). The results show that SysML configured models can be transformed in order to represent the system with functional blocks, which facilitates the generation of code.

***Keywords*:** Software Product Line, SysML, Simulink, embedded systems, Unmanned Aerial Vehicle, model transformation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ATL	*ATLAS Transformation Language*
CIM	*Computation Independent Model*
DI	*Diagram Interchange*
EMF	*Eclipse Modeling Framework*
FAA	*Federal Aviation Administration*
FB	*Functional Block*
GPS	*Global Positioning System*
IMU	*Inertial Measurement Unit*
INCT-SEC	*Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos*
M2M	*Model-to-Model*
M2T	*Model-to-Text*
MARTE	*Modeling and Analysis of Real-time and Embedded Systems*
MDA	*Model Driven Architecture*
MDD	*Model Driven Development*
MDE	*Model Driven Engineering*
MOF	*Meta-Objects Facility*
OCL	*Object Constraint Language*
OMG	*Object Management Group*
OOSEM	*Object-Oriented Systems Engineering Method*
PIM	*Platform Independent Model*
PL	*Software Product Line*
PSM	*Platform Specific Model*
SyMPLES	*SysML-based Product Line Approach for Embedded Systems*
SysML	*Systems Modeling Language*
UML	*Unified Modeling Language*
VANT	Veículo Aéreo Não-Tripulado
XMI	*XML Metadata Interchange*
XML	*eXtensible Markup Language*
YAPA	*Yet Another Paparazzi Autopilot*

SUMÁRIO

Capítulo

1

Introdução

1.1 Contextualização

Com a introdução de software em produtos de engenharia mecânica, houve um crescimento no uso de sistemas embarcados. Exemplos são veículos, equipamentos médicos, robôs, aviões entre outros. Requisitos relacionados à segurança, desempenho, utilização de recursos e confiabilidade aumentam o nível de complexidade no desenvolvimento desses sistemas (Marwedel 2010).

Sistemas embarcados são aplicações que processam informações embutidas (embarcadas) em um produto maior e que normalmente não são diretamente visíveis ao usuário (Marwedel 2010). Esse tipo de sistema se diferencia dos tradicionais por características como: heterogeneidade (hardware e software); distribuição (sobre múltiplos e heterogêneos recursos de hardware); incorporação de sensores que capturam informações do ambiente e atuadores capazes de interagir com o ambiente externo com base nas informações captadas; pouca ou nenhuma tolerância a falhas; e necessidade de tempos de resposta precisos. Por causa dessas características, os atributos de qualidade, principalmente os relacionados aos aspectos temporais são críticos.

Quando a complexidade de desenvolvimento de um sistema é alta, é comum estabelecer uma equipe multidisciplinar para dividir o sistema e tratar diversas visões. Em sistemas embarcados exemplos de visões são: hardware, software, controle e comportamental. Cada visão fornece informações relevantes a um determinado aspecto do projeto ou desenvolvimento do sistema (EL-Khoury 2006).

De acordo com Shi (2007), as visões de um sistema podem ser decompostas em áreas a serem tratadas por especialistas utilizando diversas ferramentas e seguindo padrões de projeto. Durante o desenvolvimento de um sistema embarcado, diversas linguagens de modelagem são usadas para representar os níveis de abstração necessários. Por exemplo, no desenvolvimento de sistemas automotivos, a linguagem Simulink (Simulink 1994) pode ser usada para descrever as funcionalidades do sistema de controle, enquanto a arquitetura geral do sistema pode ser descrita em *Unified Modeling Language* (UML) (UML 2005) ou por uma de suas extensões, como o *Systems Modeling Language* (SysML) (SysML 2008).

Paradigmas da engenharia de software, como Engenharia Guiada por Modelos (*Model Driven Engineering* – MDE) e Linha de Produto de Software (*Software Product Line* - PL) podem ser alternativas adequadas para auxiliar o processo de desenvolvimento de sistemas embarcados. PL permite o reuso sistemático de artefatos comuns derivados de um mesmo domínio para a construção de aplicações individuais (Linden, Schmif, e Rommes 2007), enquanto MDE apoia a geração de aplicações por meio da transformação de modelos, que podem estar no mesmo ou em diferentes níveis de abstração (Mellor 2004).

1.2 Motivação

Entre as diversas ferramentas e ambientes para o desenvolvimento de sistemas embarcados destaca-se a ferramenta de modelagem e simulação Simulink. Ela possibilita a modelagem de sistemas dinâmicos lineares, não-lineares, contínuos ou discretos no tempo. A ferramenta Simulink possui uma biblioteca de blocos funcionais, que são blocos com funções predefinidas. Os modelos são construídos por meio da composição e conexão de diferentes blocos. Além de apoiar a modelagem, simulação e teste de modelos, a ferramenta Simulink também permite a geração de código C e VHDL, automaticamente, usando o *plugin* Real-Time Workshop (Simulink Coder 2012).

O desenvolvimento de uma PL para modelos Simulink facilita a geração de produtos no desenvolvimento de sistemas embarcados. Porém, existem poucos trabalhos e ferramentas nesse contexto (Pastor, Lopez, and Royo 2006; Steiner 2012). Foi identificada apenas uma ferramenta comercial nesse domínio que é a Pure::variants (Beuche 2003). No entanto existem desvantagens das abordagens que usam a ferramenta, que são: (i) a instanciação de produtos se dá por meio de atribuição de valores a blocos de controle, o que faz com que sejam executados apenas blocos relacionados a variantes funcionais selecionadas para a instância de produto, o que aumenta a quantidade de código gerada, caso não sejam

removidos do modelo; e (ii) a representação de muitas variabilidades em um modelo Simulink aumenta a complexidade de gerenciamento com a adição de novos blocos de controle. Assim, existe a necessidade de representar variabilidades de uma PL em um alto nível de abstração. O gerenciamento da variabilidade em modelos de alto nível de abstração permite a configuração de produtos em um desenvolvimento *top-down*, sem acrescentar complexidade aos modelos de baixo nível de abstração.

Para facilitar o gerenciamento de variabilidades e permitir o desenvolvimento de sistemas embarcados a partir de modelos de alto nível foi proposta a abordagem *SysML-based Product Line Approach for Embedded Systems* (SyMPLES) (Silva 2012). SyMPLES é baseada em SysML, pois esta linguagem permite expressar conceitos importantes para a engenharia de sistemas, que não podem ser diretamente representados em linguagens de propósito geral como a UML. Silva (2012) tratou o processo de engenharia de domínio e a configuração de produtos específicos. Porém, os modelos SysML gerados pela engenharia de aplicação da PL precisam ser refinadas até chegar ao código executável. Assim, é necessário realizar a transformação do modelo SysML gerado pela PL do SyMPLES para um modelo de blocos funcionais que permite alcançar o código executável de modo automático e eficiente.

1.3 Objetivos

Considerando o contexto e a motivação apresentados nas seções anteriores, este trabalho tem o objetivo de criar um processo de transformação de modelos SysML para modelos Simulink. O processo de transformação complementa a abordagem SyMPLES proposta por Silva (2012). O processo consiste de três etapas que incluem: (i) gerar o modelo SyMPLES configurado; (ii) realizar uma transformação intermediária; e (iii) criar um script usando código Java para as interfaces de aplicação (*Application Programming Interface* - APIs) do Simulink e Stateflow (MathWorks 2012).

Modelos SysML são gerados pela fase de engenharia de aplicação a partir da arquitetura da PL. O modelo SysML usado para o processo de transformação consiste de diagramas de definição de blocos, interno de blocos e máquina de estados. Esse modelo é usado como entrada para o processo de transformação a fim de gerar o modelo Simulink com blocos funcionais e máquinas de estados Stateflow. A Figura 1 mostra uma comparação genérica desse trabalho em relação ao de Silva (2012).

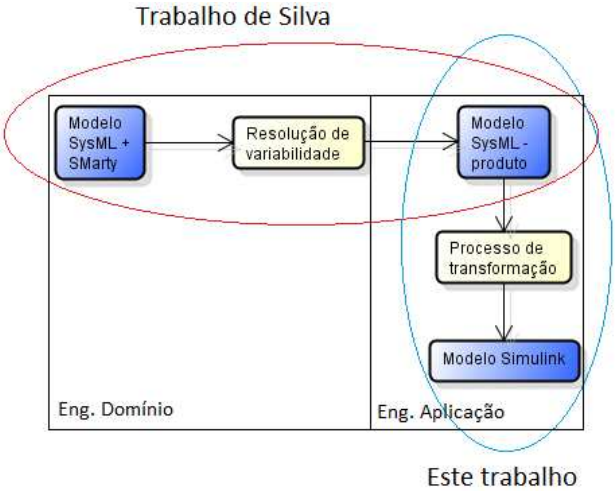

Figura 1: Relação genérica entre este trabalho e o trabalho de Silva (2012).

A avaliação do processo de transformação é realizada com base em uma PL simplificada, criada para representar uma parte do software de uma placa controladora do piloto automático Yapa 2 (YAPA 2011). A placa Yapa 2 é usada em Veículos Aéreos Não-Tripulados (VANTs) no projeto Paparazzi (Enac 2003). Foi representada uma parte do controlador de vôo do Yapa 2 em que comandos de movimento gerados pelo piloto automático são tratados para gerar comandos para os atuadores. Este trabalho está inserido no contexto do projeto vinculado ao Instituto Nacional de Ciência e Tecnologia para Sistemas Embarcados Críticos (INCT-SEC) (INCT-SEC 2012), coordenado pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC/USP).

O processo de transformação definido neste trabalho facilita a geração de modelos por meio do refinamento das abstrações. Modelos SysML configurados são usados como entrada para a transformação e representam sistemas embarcados nos níveis iniciais de desenvolvimento.

1.4 Organização

Esta dissertação está organizada em 6 (seis) capítulos. O capítulo 2 apresenta uma revisão bibliográfica sobre conceitos e desenvolvimento de sistemas embarcados. O capítulo 3 apresenta uma revisão bibliográfica sobre engenharia guiada por modelos, PL e a abordagem SyMPLES. O capítulo 4 apresenta o processo de transformação SysML para Simulink. O capítulo 5 apresenta a avaliação do processo de transformação para uma parte de um controle de vôo para a placa controladora Yapa 2. Por fim, o capítulo 6 apresenta as conclusões e os trabalhos futuros.

Capítulo

2

Sistemas embarcados

2.1. Considerações Iniciais

Este capítulo apresenta os principais conceitos sobre sistemas embarcados e engenharia de sistemas, necessários para o posterior entendimento da abordagem proposta nesta dissertação. A conceituação de sistemas embarcados, sua importância e o crescimento da complexidade do software para este tipo de sistema são apresentados inicialmente. Na sequência, é descrito o conceito de Veículos Aéreos Não-Tripulados (VANTs), pois é o principal domínio de aplicação para o qual a abordagem proposta foi aplicada. Também são apresentadas as abordagens e os principais conceitos definidos pela engenharia de sistemas para construção de sistemas embarcados, dentre eles, o método OOSEM (*Object-Oriented Systems Engineering Method*) e a linguagem SysML, utilizada como padrão para especificação de modelos de alto nível. O desenvolvimento de sistemas embarcados é descrito sob o enfoque das abordagens mais utilizadas para especificação deste tipo de sistema: a modelagem de blocos funcionais e a criação de modelos de alto nível em SysML.

2.2. Conceitos de Sistemas Embarcados

O desenvolvimento tecnológico expôs uma nova realidade: o uso intensivo pelo ser humano de sistemas computacionais. Esses sistemas, quando embutidos em um produto, são chamados de sistemas embarcados, pois constituem parte de um todo e desenvolvem tarefas específicas.

Os sistemas embarcados estão presentes em diversos setores tais como: automotivo, aeronáutico, telecomunicações, eletrônica de consumo e de dispositivos medicinais (Brisolara 2007).

Ao contrário de um sistema computacional de propósito geral, os sistemas embarcados possuem foco em uma aplicação específica. Além disso, são caracterizados por um grande número de restrições que normalmente devem ser consideradas em seus projetos, entre as quais estão (Marwedel 2010):

- Limitação de recursos computacionais: sistemas embarcados costumam possuir recursos computacionais limitados como, por exemplo, a capacidade restrita de processamento e memória, o que exige de projetistas e programadores de aplicação experiência e conhecimento sobre as tecnologias envolvidas no desenvolvimento do sistema;
- Eficiência: sistemas embarcados precisam ser eficientes. As seguintes métricas podem ser utilizadas para avaliar a eficiência de um sistema embarcado: consumo de energia, tamanho do código fonte, peso do dispositivo, custo de produção, entre outros;
- Restrições de tempo real: não completar o processamento de uma informação em determinado tempo pode resultar em perda de qualidade na avaliação do sistema, ou pode até representar risco de vida para o usuário, no caso de sistemas para trens, aviões ou da área médica;
- Custo: para projetos de sistemas embarcados com foco em altos volumes de venda (ex. eletrônicos de consumo), o mercado costuma ser bastante competitivo e para se garantir sucesso na venda de produtos é preciso haver gerenciamento eficiente do orçamento e do esforço de desenvolvimento de hardware e software.

2.2.1. Veículos Aéreos Não-Tripulados

Veículos Aéreos Não-Tripulados (VANTs) são aeronaves que podem voar sem piloto. A estrutura de aeronave possui sensores, GPS, atuadores e processadores que são combinados a um sistema computacional. Esses elementos combinados devem pilotar uma aeronave sem qualquer intervenção humana. Outra definição comum para VANT é uma aeronave capaz de voar de modo autônomo e operar em diversos tipos de missões, e que em casos de emergência pode ser controlada de alguma estação-base (Pastor, Lopez, and Royo 2006). Este foi o domínio escolhido para aplicação da abordagem SyMPLES, por ser um dos principais produtos entregáveis do INCT-SEC, projeto qual este trabalho de mestrado está inserido.

Os VANTs foram concebidos inicialmente para aplicações militares, pelo fato de trazerem soluções seguras e de relativo baixo custo para diversos tipos de missões como, por exemplo, vigilância, espionagem e bloqueio ou interferência em radares. Além das aplicações militares, essas aeronaves estão sendo cada vez mais utilizadas para aplicações civis, científicas e comerciais, como medições meteorológicas em altas altitudes, coleta de dados topográficos, avaliação de situações catastróficas e monitoração de agricultura, pesca e meio ambiente.

Um VANT opera a partir de um sistema integrado descrito por seis módulos principais que trabalham coordenadamente para alcançar autonomia de vôo e cumprir os objetivos das missões. Cada módulo é exposto a seguir, segundo Pastor, Lopez, and Royo (2006):

- Estrutura de aeronave: trata-se de uma plataforma estável, simples, leve e aerodinamicamente eficiente, com espaço limitado para aviônicos (eletrônicos de aviação);
- Computador de vôo: é o coração dos VANTs; um sistema computacional que serve para direcionar o vôo da aeronave para seu plano de vôo. Para isso ele utiliza informações aerodinâmicas coletadas por sensores (ex. acelerômetros, giroscópios, sensores de pressão e GPS), dados da missão e diversas superfícies de controle presentes na estrutura da aeronave;
- Equipamentos de coleta de dados (payload): é um conjunto de sensores que inclui câmeras, sensores infravermelhos e/ou sensores térmicos, usados para coletar informações que podem ser parcialmente processadas a bordo ou transmitidas diretamente para a estação base para que sejam analisadas futuramente;
- Controlador de missão: um sistema de computador a bordo da aeronave que controla a operação dos sensores existentes nos equipamentos de coleta de dados. Essa operação deve ser executada de acordo com o andamento do plano de vôo e com a missão atual atribuída ao VANT;
- Estação base: um sistema de computador em terra projetado para monitorar o andamento da missão e eventualmente operar o VANT e os seus equipamentos de coleta de dados;
- Infraestrutura de comunicação: uma mistura de mecanismos de comunicação (ex. modems, radio e comunicação por satélite) que devem garantir comunicação contínua entre o VANT e a estação-base.

De acordo com a FAA (2008) a combinação dos sistemas do VANT com os sistemas de apoio terrestres e o link de comunicação são chamados de sistemas de veículos aéreos não-tripulados *(Unmanned Aircraft System* - UAS). Os sistemas de apoio terrestre podem ser definidos como a estação base e outros equipamentos de suporte.

2.3. Desenvolvimento de Sistemas Embarcados

No desenvolvimento de sistemas embarcados, o uso de modelos de alto nível permite abstrair detalhes de implementação, facilitando a especificação do sistema que, inicialmente, é realizada por meio da construção de modelos ao invés de escrita de código. Com a utilização desta abordagem, modelos de sistemas embarcados podem evoluir de abstrações de alto nível até implementações, assegurando um processo mais gradativo e confiável do que o desenvolvimento com as práticas de programação tradicionais. Para tal, a linguagem de modelagem deve oferecer mecanismos para expressar não só a funcionalidade como também os requisitos da aplicação, além de apoiar a validação e mecanismos que facilitem a obtenção de uma implementação do modelo (Brisolara 2007).

2.3.1. Engenharia de Sistemas

A engenharia de sistemas é uma abordagem multidisciplinar que visa desenvolver sistemas complexos implementados por meio de soluções que envolvem hardware e software. A principal diferença entre a engenharia de sistemas e a engenharia de software, é que na primeira um sistema pode incluir componentes de hardware e de software, sendo que ambos precisam estar muito bem descritos na especificação do sistema. Na engenharia de software normalmente, o projeto de hardware no qual o sistema será executado não é especificado (Ferreira et al. 2009).

Uma visão simplificada dos processos envolvidos na engenharia de sistemas é exibida na Figura 2. O processo de especificação do sistema e projeto é usado para especificar os requisitos do sistema e dos componentes de acordo com as necessidades estabelecidas. Os componentes são então projetados, implementados e testados para assegurar sua conformidade com os requisitos. O processo de integração do sistema e teste inclui atividades para integrar os componentes ao sistema e verificar se os requisitos foram satisfeitos. Estes processos são aplicados iterativamente ao longo do desenvolvimento da aplicação, oferecendo um *feedback* contínuo entre eles. Em sistemas mais complexos, vários níveis de

decomposição podem ser aplicados nas atividades descritas anteriormente. Nestes casos, variantes do processo apresentado podem ser aplicadas recursivamente em cada nível intermediário do processo, desde o projeto e construção de componentes até a fase de validação e testes (Friedenthal, Steiner, e Moore 2008).

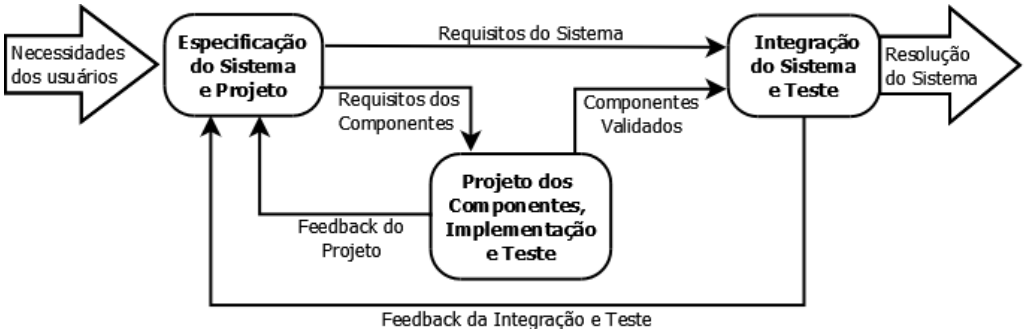

Figura 2: Esquema simplificado da abordagem de engenharia de sistemas (Friedenthal, Steiner, e Moore 2008).

Para a definição de processos, métodos e o gerenciamento de atividades, a engenharia de sistemas se baseia em uma série de padrões estabelecidos ao longo dos últimos anos. A necessidade da criação de padrões surgiu com a evolução da engenharia de sistemas e o aparecimento de um cenário em que era necessário lidar com a complexidade crescente de sistemas, principalmente das indústrias aeroespacial e de defesa. A Figura 3 mostra a taxonomia parcial dos padrões que incluem processos da engenharia de sistemas, frameworks arquiteturais, métodos, padrões de modelagem e formatos para troca de dados (Friedenthal, Steiner, e Moore 2008).

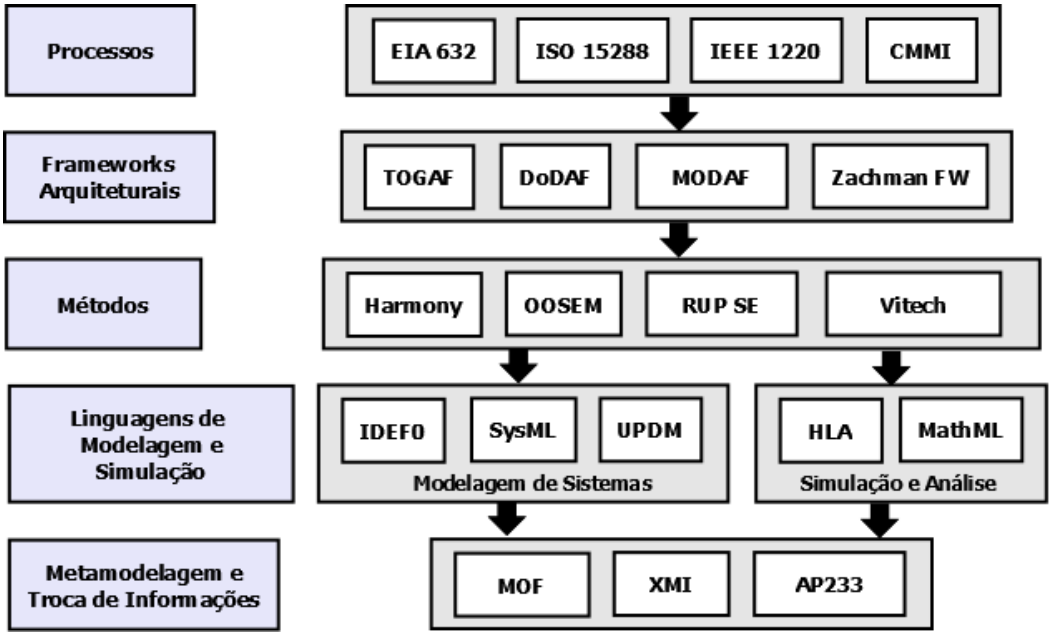

Figura 3: Taxonomia parcial dos padrões utilizados pela Engenharia de Sistemas (Friedenthal, Steiner, e Moore 2008).

Linguagens de Modelagem e Simulação é a categoria de padrões criados para a engenharia de sistemas que define uma linguagem comum para descrever os sistemas. Dentre as linguagens de modelagem destaca-se a *Systems Modeling Language* (SysML) (SysML 2008), uma extensão da linguagem UML (*Unified Modeling Language*) adotada em 2006 pela OMG (*Object Management Group*) como uma linguagem de modelagem de propósito geral para a engenharia de sistemas. SysML reutiliza vários diagramas da linguagem UML, como pode ser visualizado no diagrama da Figura 4.

SysML é uma linguagem para especificação de sistemas embarcados, que permite a modelagem de blocos ao invés de classes com o objetivo de ter um vocabulário mais próximo ao utilizado na engenharia de sistemas. SysML foi adotada no trabalho de Silva (2012) para desenvolver a abordagem SyMPLES devido ao fato dos principais processos de desenvolvimento existentes na indústria e voltados à engenharia de sistemas, como Harmony da empresa Telelogic (Bruce Powel Douglass 2008), o Object-Oriented Systems Engineering Method (OOSEM) (Lykins 2000) e o Rational Unified Process for Systems Engineering (RUP SE) (Cantor 2003), utilizarem tal linguagem para a atividade de modelagem no projeto do sistema. Esses processos possibilitam a especificação em SysML de um sistema embarcado desde a fase de elicitação de requisitos, análise e projeto, até a integração entre hardware e software, validação e testes.

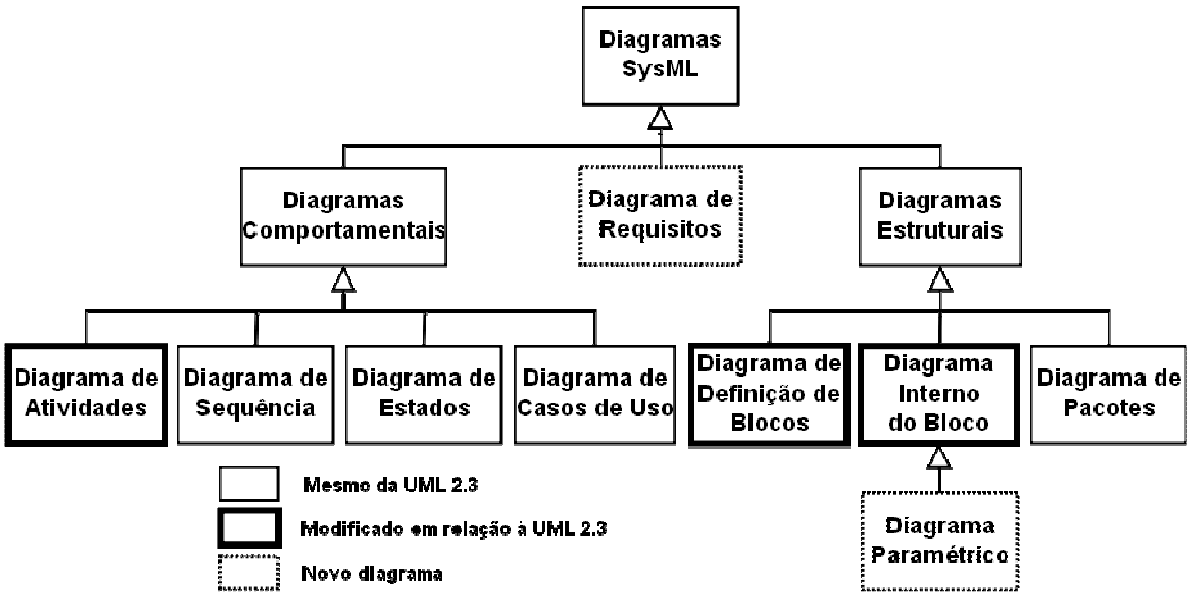

Figura 4: Taxonomia dos Diagramas SysML (Friedenthal, Steiner, e Moore 2008).

OOSEM é uma abordagem baseada em modelos que utiliza a linguagem SysML para apoiar a elicitação de requisitos, análise, projeto e verificação de sistemas. Tal abordagem utiliza os conceitos de orientação a objetos, em conjunto com os métodos *top-down* tradicionais da engenharia de sistemas, para auxiliar a criação de projetos de sistemas mais

flexíveis e extensíveis para acomodar a evolução tecnológica, como a utilização de novas ferramentas e a mudança de requisitos. A Figura 5 mostra as principais atividades do método OOSEM. Entre essas atividades estão:

- Analisar as necessidades dos usuários;
- Definir os requisitos do sistema;
- Definir a arquitetura lógica do sistema;
- Sintetizar a alocação de componentes da arquitetura;
- Validar e verificar o sistema.

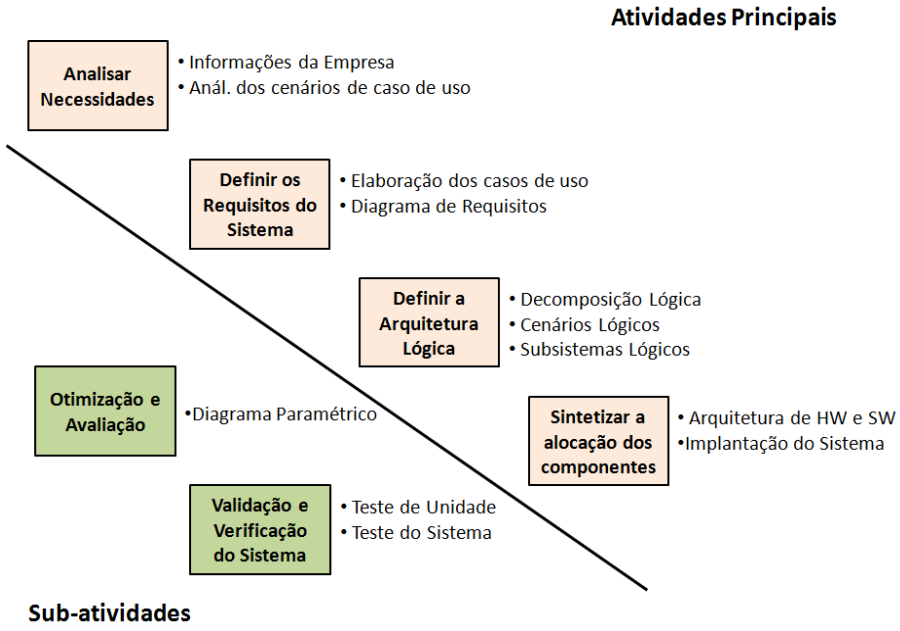

Figura 5: Atividades OOSEM (Estefan 2008).

A utilização da linguagem SysML pelo método OOSEM permite aos engenheiros especificar o sistema de forma precisa e assegurar a consistência entre as diferentes visões do sistema. Os artefatos produzidos podem ser refinados e reusados em outras aplicações, possibilitando a adoção das abordagens evolucionária e de linha de produto (INCOSE 2006).

2.3.2. Modelagem de Blocos Abstratos e Funcionais

Existem algumas abordagens para o projeto e implementação de sistemas embarcados baseadas em modelos de alto nível. Dentre elas, duas propostas se destacaram, uma baseada em blocos funcionais e outra que é baseada em orientação a objetos e provida pela linguagem UML (Bassi et al. 2011; Brisolara 2007).

A abstração é um conceito que sempre foi utilizado pelos desenvolvedores de software

para lidar com a complexidade. A linguagem UML tem sido amplamente utilizada para desenvolver e documentar sistemas tanto na academia quanto na indústria, devido a sua capacidade de extensão. A UML também tem sido aplicada em domínios que estão fora do seu escopo original, como sistemas embarcados e de tempo real (Douglass 1997).

Linguagens como SysML são utilizadas para criar modelos nas fases de requisitos, análise, projeto, validação e testes. Neste caso, torna-se vantajoso combinar SysML com os conceitos de uma linguagem orientada a blocos funcionais, pois é possível modelar um sistema desde as fases iniciais do projeto, como especificação de requisitos e análise, até a fase de implementação, por meio da utilização da abordagem de blocos funcionais. Um exemplo é o diagrama de definição de blocos SysML que é utilizado para definir blocos em termos de suas características e relacionamentos estruturais com outros blocos. A Figura 6 mostra um exemplo de diagrama de definição de blocos para uma câmera fotográfica.

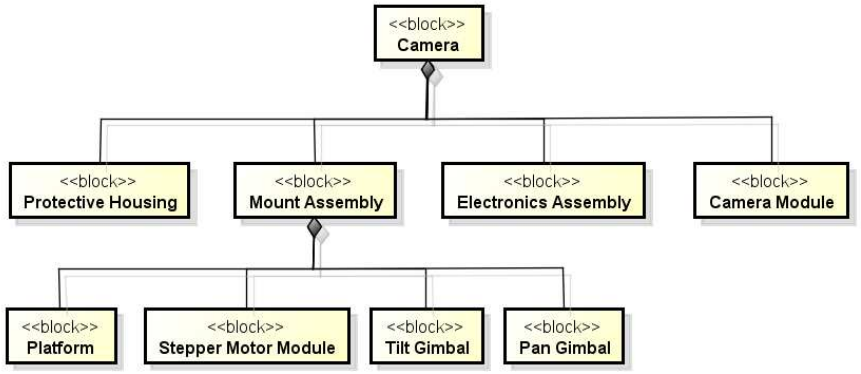

Figura 6: Diagrama de Definição de Blocos para uma câmera fotográfica (Friedenthal, Steiner, e Moore 2008).

O diagrama interno de blocos assemelha-se ao diagrama de definição de blocos, contudo, exibe as conexões entre as partes (instâncias de blocos) que compõem um determinado bloco. A Figura 7 descreve as partes da estrutura interna dos blocos *Camera Module* e *Electronics Assembly*, mostrados na Figura 6.

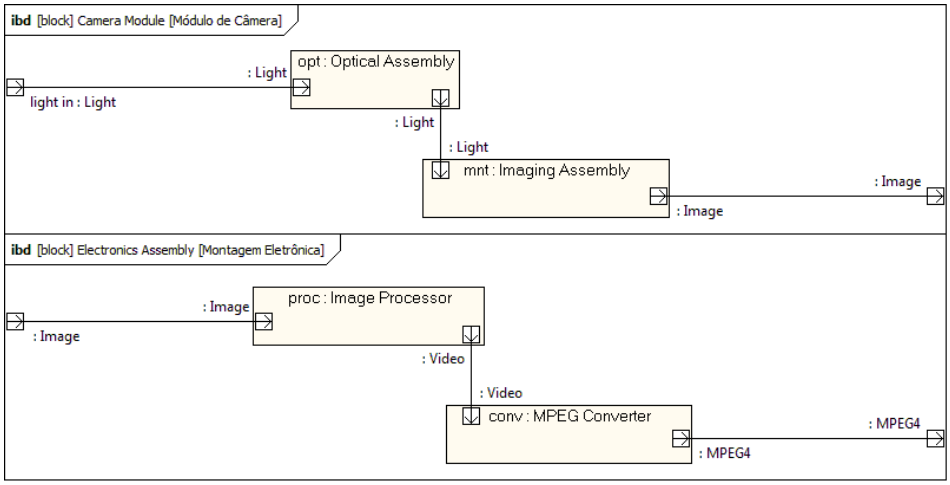

Figura 7: Diagrama Interno de Blocos (Friedenthal, Steiner, e Moore 2008).

Na modelagem de Blocos Funcionais (BFs), as aplicações são projetadas a partir da conexão de diversos BFs. Esta abordagem não possui mecanismos para expressar os requisitos do sistema, pois a modelagem se inicia com a construção da solução em mais baixo nível para o problema que está sendo considerado. A comunicação entre os blocos ocorre por meio do fluxo de dados entre as portas ao invés de troca de mensagens, como acontece na abordagem de orientação a objetos. O comportamento de cada bloco é descrito utilizando linguagens orientadas a blocos funcionais, como linguagens para controladores programáveis (LCPs) ou Simulink. Ferramentas disponíveis na indústria para este tipo de aplicação também utilizam blocos funcionais como unidade de organização. Apesar de algumas diferenças, o conceito de bloco funcional é o mesmo em grande parte destas linguagens e ferramentas (Heverhagen 2003).

Os elementos básicos de uma linguagem de blocos funcionais são: blocos, portas e conectores. Um bloco representa uma entidade; blocos podem conter outros blocos, representando o conceito de subsistema. Um conector liga dois blocos por meio de portas. Os conectores representam a interação das entidades do modelo em termos de transmissão de dados e controle de fluxo (Sjöstedt et al. 2008). A Figura 8 mostra um exemplo de sistema que representa um loop simples formado por oito blocos conectados, utilizando blocos funcionais da linguagem Matlab/Simulink.

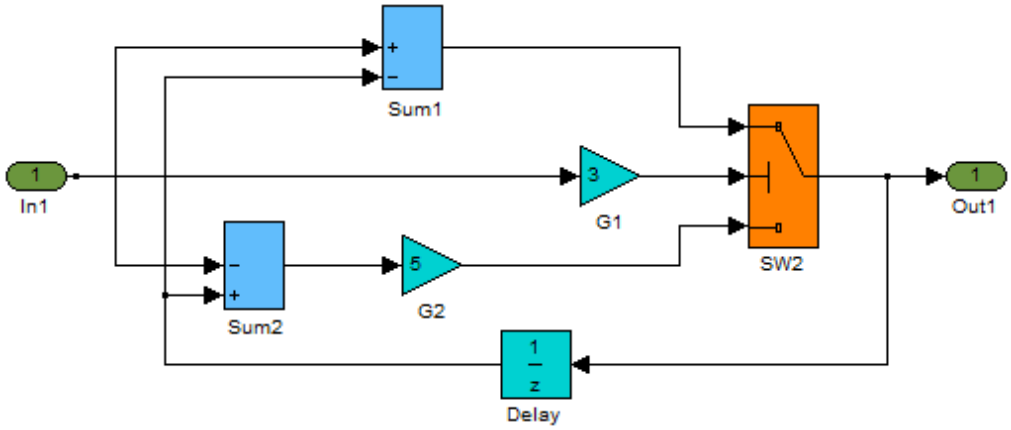

Figura 8: Exemplo de sistema como um modelo Simulink (Simulink 1994).

Nas linguagens de blocos funcionais, o comportamento de um bloco é pré-definido. Por exemplo: na Figura 8, os blocos G1 e G2 são do tipo *Gain*. Esse tipo de bloco tem como função gerar um valor igual à entrada multiplicada por um parâmetro definido pelo desenvolvedor, no caso desse exemplo, `G1` tem um ganho de 3x e o `G2` de 5x.

2.4. Considerações Finais

Este capítulo apresentou uma revisão bibliográfica necessária para o entendimento desta dissertação. Sistemas embarcados estão cada vez mais presentes no cotidiano das pessoas, desde eletrônicos de consumo até sistemas que controlam o tráfego de trens e aviões. A evolução das plataformas de hardware possibilitou mover mais funcionalidade para o software, o que aumentou a complexidade deste tipo de sistema. Neste cenário, a construção de modelos de alto nível tem sido adotada para lidar com a complexidade das novas gerações de sistemas embarcados. Uma das principais linguagens utilizadas para construção de modelos em nível mais alto de abstração para sistemas embarcados é a UML. Contudo, tal linguagem é de propósito geral e não considera alguns aspectos relacionados ao desenvolvimento de sistemas embarcados. Neste sentido, a construção de modelos em um ambiente apoiado pela engenharia de sistemas possibilita a utilização de métodos e padrões apropriados para a especificação de sistemas embarcados.

Capítulo

3

Engenharia Guiada por Modelos

3.1. Considerações Iniciais

Este capítulo apresenta os principais conceitos sobre engenharia guiada por modelos, necessários para o posterior entendimento da abordagem proposta nesta dissertação. Inicialmente são apresentados os fundamentos básicos e padrões relacionados à engenharia guiada por modelos. Na sequência, é apresentada a relação entre modelos, metamodelos e plataformas. Em seguida, são apresentados os conceitos sobre transformação de modelos com seus tipos, categorias, pontos de vista e linguagens de transformação. Por fim, são apresentados os conceitos iniciais sobre linha de produto de software e a abordagem SyMPLES complementada neste trabalho.

3.2. Introdução a Engenharia Guiada por Modelos

A Engenharia Guiada por Modelos (*Model-Driven Engineering* - MDE) é uma abordagem que propõe o uso de técnicas transformacionais em projetos de sistemas, para que a fase de implementação seja (semi-)automaticamente derivada da especificação de modelos. Nesta abordagem, modelos são utilizados como o principal artefato ao longo do ciclo de vida de desenvolvimento do software (Selic 2003). A MDE combina dois conceitos:

- **Linguagens Específicas de Domínio (*Domain Specific Languages* - DSLs)** que formalizam a estrutura, comportamento e os requisitos de um domínio em particular. Além disso, definem o relacionamento entre os principais conceitos do domínio, assim

como especificam as restrições dos conceitos chaves e a semântica relacionada a eles. DSLs são descritas em termos de metamodelos, cujos elementos representam conceitos do domínio. Instâncias dos metamodelos representam a utilização dos conceitos do domínio em um projeto;

- **Mecanismos de Transformação e Geradores** cujo propósito é interpretar a informação contida no modelo para produzir (semi-)automaticamente outros tipos de artefatos, como modelos mais detalhados, código-fonte, arquivos de configuração, entre outros. Ferramentas podem ser utilizadas para auxiliar a manter a consistência entre a especificação do sistema e sua implementação.

Existem diversas iniciativas guiadas por modelos (*Model-driven*) com novas terminologias que tem um significado particular (Mellor 2004). As iniciativas (vide Figura 9) são:

- **Arquitetura baseada em modelos (*Model-Driven Architecture - MDA*):** a *Object Management Group* (OMG) propôs alguns padrões que especificam a interoperabilidade entre tecnologias no desenvolvimento dirigido por modelos com transformações automáticas. Porém, qualquer linguagem usada em MDA precisa ser descrita pelos termos definidos pelo padrão MOF, que permite o entendimento dos metadados como uma precondição para qualquer transformação automática;
- **Desenvolvimento baseado em modelos (*Model-Driven Development - MDD*):** de acordo com Mellor, Clark, & Futagami (2003) o objetivo do MDD é simplificar a notação que é usada para construir o modelo de um sistema e poder transformar em algo real. A diferença entre MDD e MDA é que MDD não está preso aos padrões da OMG oferecendo flexibilidade para o desenvolvedor;
- **Engenharia guiada por modelos (*Model-Driven Engineering - MDE*):** de acordo com Schmidt (2006), a MDE é um paradigma de engenharia de software que lida com a complexidade das plataformas, assim é capaz de expressar eficientemente os conceitos específicos de domínio.

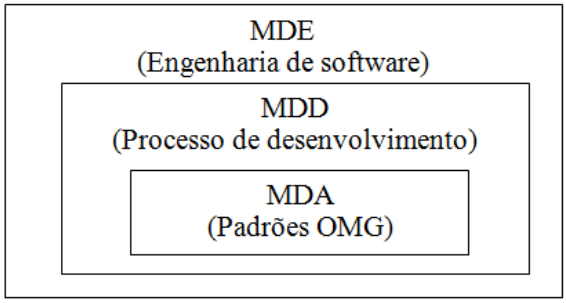

Figura 9: Representação das iniciativas guiadas por modelos.

3.3. Model Driven Architecture

Um exemplo padrão de MDE é a *Model Driven Architecture* (MDA) (Miller and Mukerji 2003) proposta pelo *Object Management Group* (OMG). Os elementos do MDA são descritos a seguir:

- **Meta-Objects Facility (MOF)** - um padrão OMG para especificação de metamodelos;
- **Unified Modeling Language (UML)** - uma linguagem de modelagem de propósito geral para a especificação de sistemas. Ela foi definida com base no padrão MOF e representa a linguagem padrão para modelagem de sistemas;
- **MOF Query/View/Transformation (QVT)** - define os requisitos para linguagens de transformação e as regras de mapeamento necessárias para permitir a transformação entre modelos de acordo com os metamodelos definidos no padrão MOF;
- **XML Metadada Interchange (XMI)** - padrão para troca de informações de metadados, especificados em XML, que permite a troca de informações sobre especificações baseadas em MOF, como por exemplo, a troca de modelos UML entre diferentes ferramentas.

Para representar visualmente o padrão MDA, a OMG estabeleceu um *framework* conceitual baseado em um conjunto de camadas e transformações. Os três níveis de modelo (vide Figura 10) são:

- O modelo independente de computação, denominado CIM (*Computation Independent Model):* o objetivo de um CIM é atuar como um contrato que funciona como referência para verificar se os requisitos do cliente estão corretamente especificados, descrevendo a funcionalidade dos negócios da aplicação e seu comportamento, por meio de diagramas de caso de uso e diagramas de atividades, definindo inclusive os atores que interagem com a aplicação;

- O modelo independente de plataforma, denominado PIM (*Platform Independent Model):* um PIM pode ser definido como uma visão do sistema sem qualquer especificação dos detalhes de implementação, para não estabelecer dependências de plataforma (software ou hardware). Contudo, o PIM necessita que seus modelos contenham informação suficiente da lógica de programação, para que o código possa ser concretamente gerado a partir de modelos como diagramas de classe (estrutura) e diagramas de sequência (comportamento); e,
- O modelo específico de plataforma, denominado PSM (*Platform Specific Model):* o objetivo deste modelo é facilitar a geração de código a partir do modelo PIM, incluindo informações para execução em uma determinada plataforma (software ou hardware). O padrão MDA sugere a utilização de perfis UML para criar modelos contendo código para uma plataforma específica.

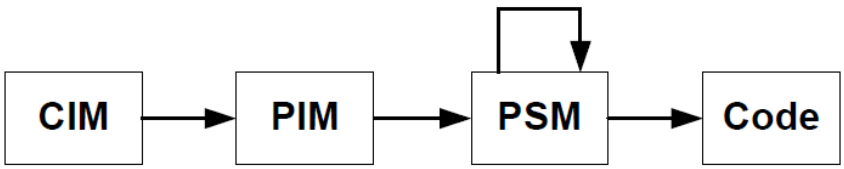

Figura 10: Modelos e transformações em MDA.

3.4. Modelos, Metamodelos e Independência de Plataforma

Na computação e no desenvolvimento de sistemas, uma definição consensual de modelo é dada por Rothenbertg (1989):

> "*...Um modelo representa a realidade para um determinado propósito; um modelo é a abstração da realidade no sentido de que ele não pode representar todos os aspectos da realidade. Isso nos permite lidar com o mundo de uma forma mais simplificada, evitando a complexidade da realidade...*"

Para cada ciclo da fase de desenvolvimento de um software, o padrão MDA sugere a elaboração de um modelo correspondente, por meio de uma notação bem definida (Miller and Mukerji 2003). Modelos usados no MDA são expressos em UML. Contudo, analisar, automatizar e transformar modelos requer uma forma não-ambígua de descrever a semântica do modelo. Esta habilidade pode ser obtida com a utilização de metamodelos que definem a estrutura, a semântica e as restrições que devem ser levadas em consideração para a construção de determinados modelos. Por exemplo, o metamodelo UML especifica que

modelos podem conter pacotes, pacotes podem conter classes e classes podem conter atributos e operações. A Figura 11 mostra uma parte do metamodelo de classes UML com as relações entre alguns elementos relacionados a uma classe.

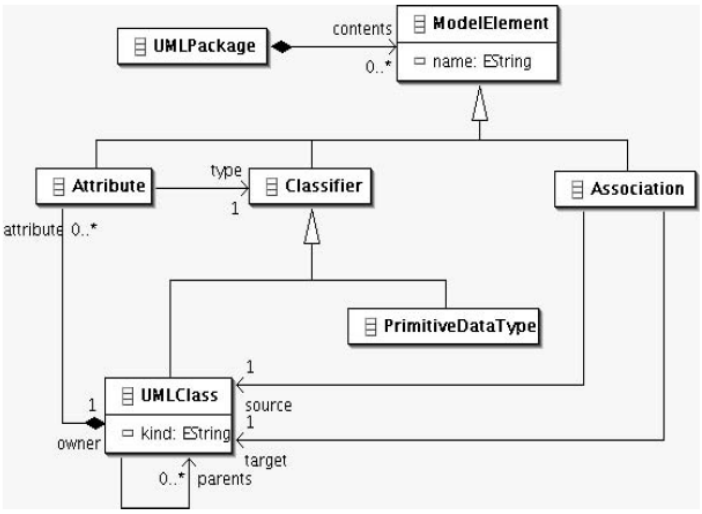

Figura 11: Parte do metamodelo de classes UML (Mellor, 2004).

Um metamodelo é descrito por um meta-metamodelo. Atkinson and Kuhne (2003) definem uma arquitetura de metamodelagem em quatro camadas (vide Figura 12) que mostra o relacionamento entre meta-metamodelos, metamodelos e modelos e uso do padrão MOF usado para representar modelos UML. As camadas de metamodelagem são:

- M3: camada que contém meta-meta-metadados para descrever as propriedades que o os meta-metadados podem exibir.
- M2: camada que contem meta-metadados para descrever as propriedades que os metadados podem assumir. Para UML esses elementos seriam classes, atributos, operações, etc.
- M1: camada que contem os metadados da aplicação. Exemplos são classes de um sistema orientado a objetos ou uma tabela de definições de um banco de dados relacional;
- M0: camada que contém os dados da aplicação. Exemplos são instancias das classes em tempo de execução ou linhas de tabelas de bancos de dados relacionais.

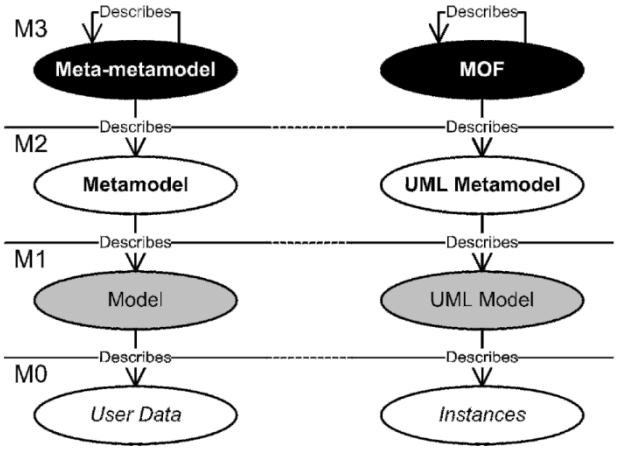

Figura 12: Arquitetura de metamodelagem em quatro camadas (Atkinson and Kuhne 2003).

O padrão MDA se baseia na independência de plataforma, mas um metamodelo também pode descrever algumas características de uma plataforma específica, e por sua vez, uma plataforma específica pode ser descrita por mais de um metamodelo. Mellor et al. (2003) descrevem o conceito de plataforma como a especificação de um ambiente de execução para um determinado conjunto de modelos. O relacionamento entre modelos, metamodelos e plataformas pode ser visualizado na Figura 13.

Figura 13: Relacionamento entre modelos, metamodelos e plataformas (Almeida 2008).

3.5. Transformação de Modelos

A transformação de modelos descreve um processo em que um modelo é transformado em outro modelo de acordo com seus metamodelos (Czarnecki e Helsen 2003). No processo de transformação é descrito o relacionamento entre modelos, mais especificamente o mapeamento de informação entre os modelos a partir de regras de mapeamento. As regras podem ser interpretadas por uma máquina de transformação de modelos. A transformação de modelos envolve dois modelos: o modelo fonte e o modelo alvo. Esses dois modelos podem ser descritos por diferentes metamodelos.

A Figura 14 mostra os conceitos básicos de uma transformação de modelos. Um modelo fonte (*Source model*) é transformado em um modelo alvo (*Target model*) que são descritos por metamodelos (*Metamodel*). As regras de mapeamento (*Transformation rules*) são descritas por uma linguagem de transformação (*Transformation language*) e podem ser executadas por uma máquina de transformação (*Transformation Engine*). A linguagem de

transformação é descrita pelo padrão MOF (Ruscio 2007).

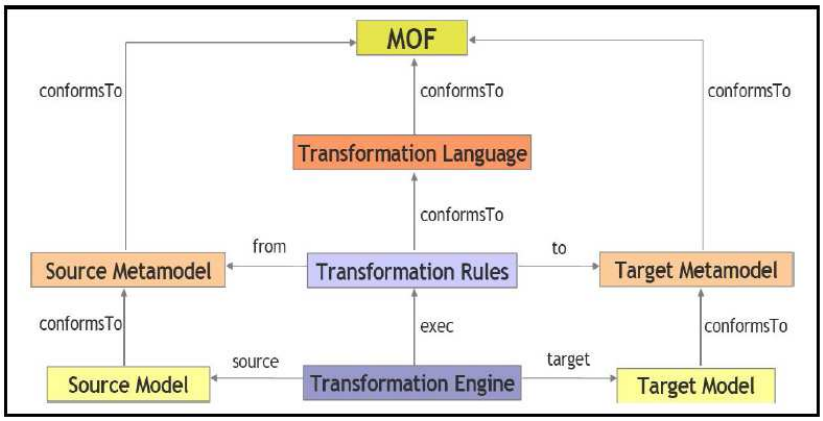

Figura 14: Transformação de modelos baseado em metamodelos (Ruscio 2007).

3.5.1. Tipos de Transformação de Modelos

O processo de transformação de modelos envolve três tipos de transformações (vide Figura 15): (i) as transformações de modelos em diferentes níveis de abstração; (ii) as transformações baseadas em engenharia reversa; e (iii) refinamentos de modelos no mesmo nível de abstração. As transformações de modelos do tipo refinamento permitem a criação de visões heterogêneas. Um exemplo de modelo PIM que pode ser refinado são diagramas estruturais e/ou comportamentais representados em UML.

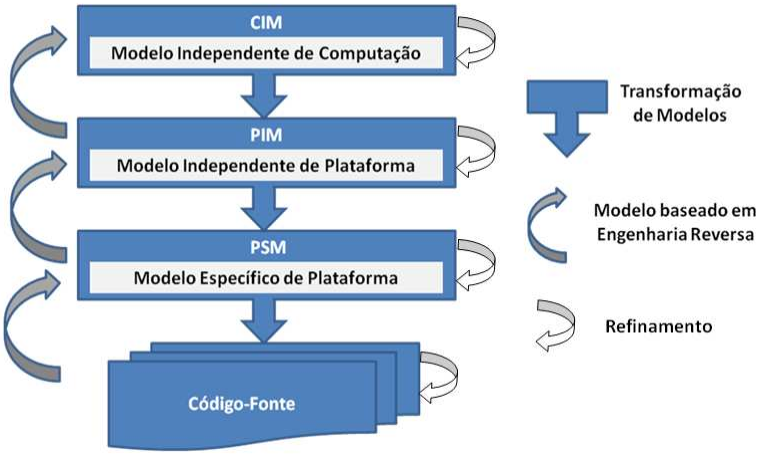

Figura 15: O processo de transformação de modelos da MDA (Almeida 2008).

3.5.2. Categorias de Transformação de Modelos

Diversas técnicas de transformação foram elaboradas para diversos domínios. Diversos autores como Kleppe, Warmer, & Bast (2003) classificam transformadores de modelos em

duas categorias: (i) M2M - Modelo-para-Modelo (*Model-to-Model*) que representa a transformação entre modelos; e (ii) M2T - Modelo-para-Texto (*Model-to-Text*) que representa a transformação de um modelo para algum código ou texto específico. A metodologia normalmente usada em MDA é descrita como um processo de transformação como mostra a Figura 16.

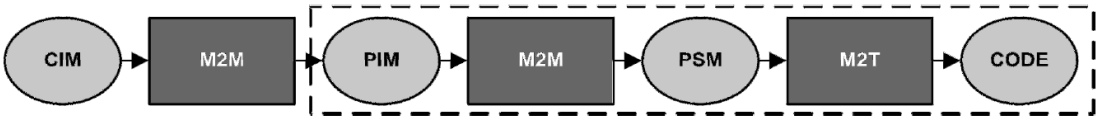

Figura 16: Fluxo de transformação MDA (Kleppe, Warmer, and Bast 2003).

3.5.3. Pontos de Vista de Transformação de Modelos

Existem dois pontos de vista para a transformação de modelos chamados: (i) Translationist – as transformações entre os modelos PIM, PSM e código acontecem apenas em uma direção (abstrato para código). Essa abordagem é defendida no livro de Mellor & Balcer (2002) e usada em trabalho como o de Sjöstedt et al. (2008); e (ii) Elaborationist – as transformações podem acontecer nos dois sentidos permitindo que os modelos mais abstratos sejam atualizados caso o código seja alterado (usando ferramentas de integração). Essa abordagem é defendida no livro de Kleppe, Warmer, & Bast (2003) e usada em trabalhos como o de Biehl, Sjöstedt, e Törngren (2010). A Figura 17 mostra os dois pontos de vista de desenvolvimento.

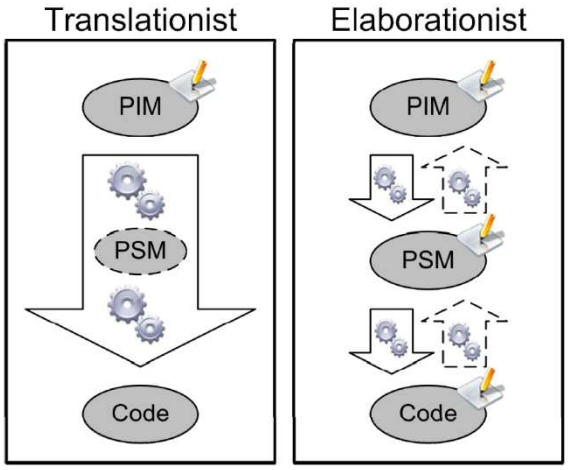

Figura 17: Abordagens de transformação entre modelos.

3.5.4. Linguagem de Transformação ATL

As linguagens de transformação de modelos são usadas para representar as regras de transformação que são executadas por máquinas de transformação. *Atlas Transformation*

Language (ATL) (Obeo 2006) é a linguagem de transformação de modelos mais popular, pois pode manipular metamodelos na plataforma do *Eclipse Modeling Framework* (EMF) (EMF 2012). O EMF é um *framework* de geração de código e modelagem que auxilia na criação de aplicações dirigidas por modelo. O EMF usa o meta-metamodelo padrão Ecore que é baseado no MOF. De acordo com Louhichi et al. (2011) o EMF é a plataforma MDE mais utilizada.

ATL é uma linguagem hibrida contendo a mistura de construtores declarativos e imperativos baseado na linguagem de restrições *Object Constraint Language* (OCL) para escrever expressões. As transformações ATL são unidirecionais, assim modelos fonte podem ser lidos e navegados, porém não alterados, enquanto modelos alvo não podem ser navegados (Louhichi et al. 2011).

A linguagem ATL representa as regras de transformação de modelos da categoria M2M. Os elementos de um arquivo de regras ATL são:

- ***Headers:*** definem os atributos relativos à transformação. Por exemplo, o(s) metamodelos de entrada e de saída;
- ***Import:*** seção opcional usada para importar bibliotecas ATL;
- ***Helpers:*** funções ATL semelhantes à linguagem Java;
- ***Rules***: regras de transformação baseadas nos metamodelos. Existem 2 tipos de regras: (i) construtores declarativos que representam mapeamentos simples; e (ii) construtores imperativos que representam mapeamentos complexos chamados *Called rules*.

Os construtores declarativos são divididos em 3 tipos:

- ***Matched rules:*** são aplicadas uma vez para cada correspondência e são executadas sequencialmente;
- ***Lazy rules***: são aplicadas varias vezes para cada correspondência, porém precisam ser referenciadas por outras regras para executar;
- ***Unique lazy rules***: é aplicada uma vez a cada correspondência e necessita ser referenciada para ser executada.

A Figura 18 mostra um exemplo de regra do tipo matched rule de livro (`book`) para publicação (`publication`). Cada regra tem um nome único, as atribuições são determinadas pelo símbolo <– e são separados por vírgulas. A regra representa que apenas livros com mais de 2 páginas são considerados publicações.

```
rule Book2Publication {
	from
		b : Book!Book (
			b.getNbPages() > 2
		)
	to
		out : Publication!Publication (
			title <- b.title,
			authors <- b.getAuthors(),
			nbPages <- b.getNbPages()
		)
}
```

Figura 18: Exemplo de *matched rule* ATL.

3.6. Linha de Produto de Software

O software para sistemas embarcados tem evoluído e aumentado o grau de complexidade. Requisitos como extensibilidade e customização para clientes específicos tem se tornado cada vez mais frequentes, e o tempo de entrega exigido para o produto vem diminuindo. Para lidar com estas necessidades, algumas abordagens da engenharia de software têm emergido como a linha de produto e a MDE.

Linha de Produto de Software (*Software Product Line - PL*) é uma abordagem usada no desenvolvimento de software, que desenvolve uma família de produtos com foco em um domínio específico. Os produtos de uma mesma família possuem uma arquitetura reusável comum, uma vez que são destinados a um mesmo domínio, mas também possuem características variáveis para satisfazer a diferentes requisitos específicos. Os diferentes produtos de uma PL para sistemas embarcados podem variar em termos de comportamento, atributos de qualidade, plataforma, configuração física, *middleware*, entre outros aspectos (Linden, Schmif, e Rommes 2007).

Ao contrário do desenvolvimento de sistemas únicos, no desenvolvimento de uma PL, seus requisitos, projeto e casos de teste devem ser considerados para todo o domínio. As principais atividades de uma PL são: (i) engenharia de domínio no qual o núcleo da infraestrutura é desenvolvido pela representação explícita de pontos de variação (do inglês *Variation Points*) que permitem a configuração de produtos; e (ii) a engenharia de aplicação no qual os pontos de variação são resolvidos a partir do núcleo da arquitetura. Modelos de características são usados para capturar e gerenciar similaridades e variabilidades da arquitetura de uma PL. Esse modelo é criado na fase de engenharia de domínio e usado como entrada para a engenharia de aplicação. Um modelo de características também pode ter

características obrigatórias, opcionais e alternativas (Linden, Schmif, e Rommes 2007).

A abordagem de PL tem sido importante no domínio de sistemas embarcados (Braga et al. 2011; Fragal, Oliveira, e Gimenes 2011; Polzer, Kowalewski, e Botterweck 2009), pois os requisitos de aplicações específicas geralmente causam mudanças no núcleo da arquitetura de software. Assim, o desenvolvimento de uma PL pode facilitar a evolução sistemática dos produtos e reduzir os custos de desenvolvimento.

As abordagens de MDE e PL fornecem uma infraestrutura em que produtos diferentes, porém relacionados, podem ser obtidos. Os benefícios de aplicar as duas abordagens separadamente são bastante conhecidos na indústria. Porém, a natureza complementar das duas técnicas permite a combinação de ambas em uma abordagem que pode ser chamada de Linha de Produto Guiada por Modelos (*Model Driven Product Line* - MDPL) (Czarnecki et al. 2005).

3.7. A Abordagem SyMPLES

A abordagem SyMPLES é derivada do método OOSEM (Lykins 2000) usado na engenharia de sistemas e também é derivado da abordagem SMarty (Junior, Gimenes, e Maldonado 2010) usada na representação de variabilidades para UML. SyMPLES reúne duas técnicas que têm sido utilizadas para lidar com a crescente complexidade da atual geração de sistemas embarcados: PL e os conceitos de orientação a objetos aplicados na construção de modelos de alto nível. SyMPLES é composta de dois perfis, que são extensões para a linguagem SysML, criadas por meio do mecanismo de *profiling*; e dois processos conforme descritos a seguir.

PERFIS:

- SyMPLES Profile for Functional Blocks (SyMPLES-ProfileFB) é um perfil para blocos funcionais, formado por um grupo de estereótipos cujo objetivo é fornecer uma semântica adicional aos blocos SysML. Nesse sentido, é possível associar um determinado tipo de comportamento a um bloco padrão SysML, para facilitar a especificação de sistemas embarcados;
- SyMPLES Profile for Representation of Variability (SyMPLES-ProfileVar) é um perfil para representação de variabilidades, baseado no perfil UML definido na abordagem SMarty (Junior, Gimenes, e Maldonado 2010). A partir de um conjunto de estereótipos e valores etiquetados (*tagged values*), o SyMPLES-ProfileVar possibilita a representação dos artefatos de uma PL para sistemas embarcados baseada em SysML, permitindo a especificação dos aspectos estruturais, comportamentais e de

variabilidade em uma única notação.

PROCESSOS:

Cada um dos processos é formado por um grupo de atividades e um conjunto de diretrizes para cada atividade, que sistematiza a execução do processo conforme descrito a seguir:

- SyMPLES Process for Product Lines (SyMPLES-ProcessPL) é um processo de desenvolvimento de PL que define um conjunto de atividades genéricas, independentes de ferramentas, que auxiliam a criação dos artefatos de uma PL para sistemas embarcados baseada em SysML;
- SyMPLES Process for Identification of Variabilities (SyMPLES-ProcessVar) é um processo para representação das variabilidades de uma PL, baseado no processo definido na abordagem SMarty. O SyMPLES-ProcessVar é executado paralelamente ao processo SyMPLES-ProcessPL; seu objetivo é auxiliar o usuário na identificação, delimitação e representação das variabilidades, além da configuração de produtos de uma PL para sistemas embarcados baseada em SysML.

A abordagem SyMPLES se concentra na fase de engenharia de domínio de uma PL. Seu objetivo é definir uma PL para sistemas embarcados que permita a geração de produtos a partir de modelos. Com isso, é possível expressar os conceitos relacionados à estrutura, comportamento, definição e resolução de variabilidades em SysML.

3.8. Considerações Finais

Este capítulo apresentou os conceitos relacionados à engenharia guiada por modelos (MDE), gerenciamento e representação de variabilidades de PL, e a abordagem SyMPLES necessários para o entendimento da proposta desta dissertação. MDE é uma abordagem promissora para o desenvolvimento de software, pois permite a trasnformação de modelos que podem evoluir de abstrações de alto nível até implementações. A abordagem de PL também pode ser aplicada no desenvolvimento de sistemas embarcados em conjunto com a MDE. Também foi apresentado a abordagem SyMPLES que usa conceitos de PL para sistemas embarcados e é extendida neste trabalho com o uso de técnicas de MDE.

Capítulo

4

Transformação SysML para Simulink na Engenharia de Aplicação

4.1 Considerações Iniciais

Este capítulo apresenta um extensão da abordagem *SyMPLES*, proposta para o desenvolvimento de PLs baseadas em SysML para o domínio de sistemas embarcados. Esta extensão refere-se ao desenvolvimento da atividade de engenharia de aplicação, principalmente a transformação de modelos SysML para Simulink. Inicialmente são apresentadas as ferramentas usadas para executar a transformação dos modelos. Em seguida, é apresentada a relação entre a abordagem SyMPLES e o processo de transformação. Em seguida é apresentada a extensão do perfil *SyMPLES Profile for Functional Blocks* (*SyMPLES-ProfileFB*), que fornece uma semântica adicional aos blocos SysML e auxilia na especificação de sistemas embarcados. Em seguida, são apresentadas as etapas do processo de transformação detalhadas com um exemplo de aplicação.

4.2 Ferramentas Utilizadas

SyMPLES representa a arquitetura da PL em SysML e utiliza como apoio para a representação dos modelos a ferramenta TOPCASED (*Toolkit in Open source for Critical Applications & SystEms Development*) (TOPCASED 2012). As variabilidades da PL são

representadas e resolvidas com o apoio da ferramenta Pure::variants (Beuche 2003). A ferramenta de apoio chamada *SysML Importer* (Silva, 2012) faz a leitura das variabilidades presentes no modelo SysML e as importa como um projeto para a Pure::variants. Assim, após a resolução das variabilidades são geradas especificações de produtos da PL em SysML.

Na versão do ambiente 5.2 do TOPCASED (TOPCASED 2012) os editores SysML da ferramenta Papyrus (Papyrus 2012) foram integrados. O Papyrus e TOPCASED têm editores gráficos baseados em metamodelos Ecore que são usados para representação de diagramas SysML. Editores gráficos baseados no Ecore usam dois arquivos para armazenar os elementos modelados:

- Um arquivo é chamado de modelo de domínio. Esse arquivo armazena dados gerados de elementos UML (nome, tipo e relacionamentos) e os estereótipos dos perfis aplicados ao modelo;
- O outro arquivo armazena informações gráficas do editor gráfico. Por exemplo, o metamodelo *Notation* é um padrão usado pelo *Graphical Modeling Framework* (GMF) (Hunter 2012) baseado no EMF para representação de informações de diagramas separado do modelo de domínio. Dados como posição e tamanho dos elementos representados no editor gráfico são armazenados nesse arquivo. Ele é usado como base para o padrão *Diagram Interchange* (DI) (OMG 2006) usado pelos editores da ferramenta *Papyrus*.

Para representar os modelos SysML foi usado o editor gráfico Papyrus integrado ao ambiente TOPCASED. O editor Papyrus apresenta melhores recursos para a representação dos diagramas SysML que o editor gráfico básico TOPCASED. Porém, ambos são baseados no framework EMF.

A ferramenta usada para representar blocos funcionais é a Simulink. Ela é muito usada na indústria e tem uma API que permite a geração de modelos.

4.3 Processo de Transformação SysML para Simulink

A transformação de um produto gerado pela PL para um modelo de blocos funcionais não foi tratada pela abordagem SyMPLES, assim um processo de transformação é necessário para realizar tal tarefa para viabilizar o refinamento dos produtos gerados. Portanto, o trabalho de Silva (2012) contempla a engenharia de domínio da PL enquanto a ênfase deste trabalho está na transformação de modelos na engenharia de aplicação da PL. A Figura 19 mostra uma visão geral da relação entre esse trabalho e a abordagem SyMPLES para tratar com a

transformação de modelos SysML para Simulink. O refinamento de um modelo SysML com variabilidades representadas por estereótipos do SyMPLES-ProfileVar para a especificação de um modelo SysML é a primeira transformação entre modelos PIM. A segunda transformação entre modelos PIM ocorre a partir do produto especificado em SysML gerado pela PL para um modelo Simulink.

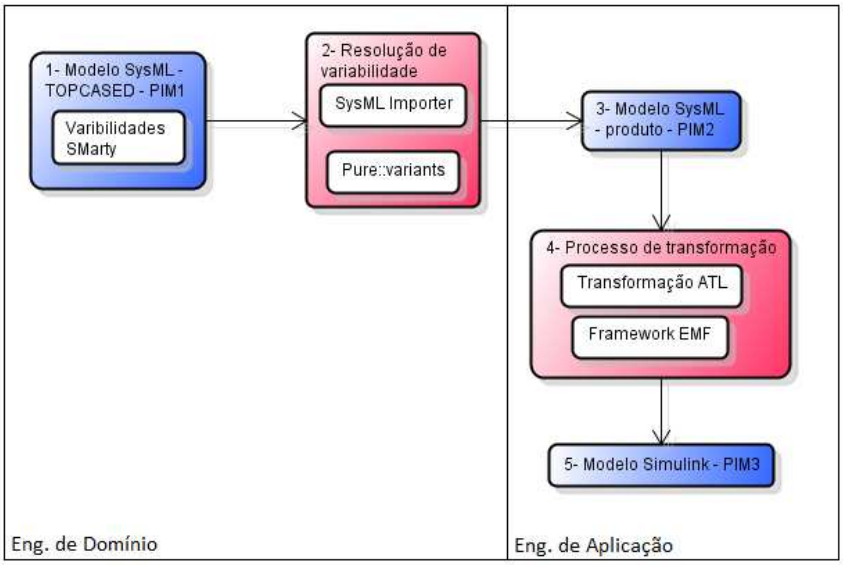

Figura 19: Representação genérica da extensão SyMPLES.

O processo de transformação utiliza o framework EMF (EMF 2012) em conjunto com a linguagem ATL (Obeo 2006) usada para definir as regras de transformação. Os arquivos XMI que armazenam os dados gráficos e de domínio SysML contém muitas informações não mapeadas e desnecessárias para a transformação. A linguagem ATL é usada para realizar uma transformação intermediária que torna o processo flexível para tratar outros editores SysML baseados no EMF.

Diagramas estruturais e comportamentais de SysML são usados na transformação para o modelo Simulink conforme mostra a Figura 20. Os diagramas estruturais que serão usados na transformação são os diagramas de definição de blocos e diagramas internos de blocos. O diagrama comportamental usado na transformação é o diagrama de estados.

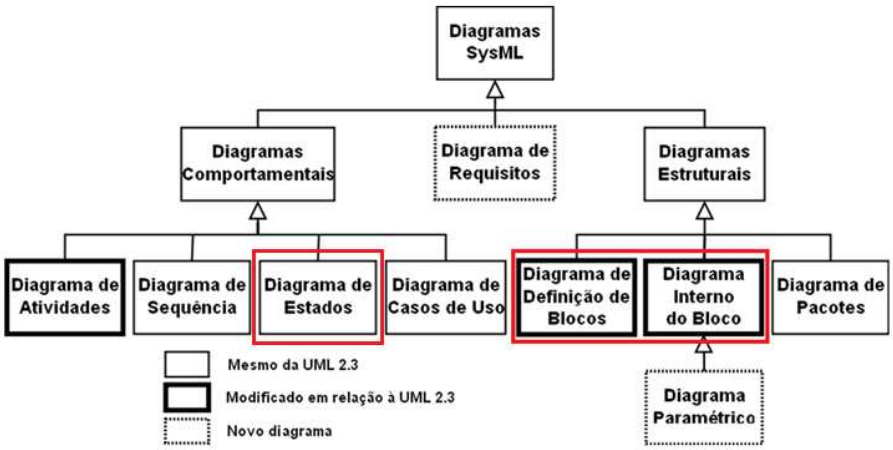

Figura 20: Diagramas SysML usados na transformação para Simulink.

SyMPLES apresenta um método para elaborar a arquitetura de uma PL para sistemas embarcados nos níveis iniciais de desenvolvimento. Os modelos representados em SysML usam também estereótipos definidos pelo SyMPLES-ProfileFB. Os estereótipos permitem o mapeamento de blocos SysML para blocos funcionais. O modelo SysML gerado na engenharia de aplicação da PL pode ser transformado em um modelo de blocos funcionais. Para permitir a transformação de modelos SysML para blocos funcionais Simulink, foi desenvolvido um processo de transformação que estende a abordagem SyMPLES. O processo de transformação tem como objetivo facilitar a geração de código por meio da transformação de modelos como descrita na abordagem MDE. As seções seguintes apresentam inicialmente a extensão SyMPLES-ProfileFB necessária para o processo de transformação seguida das etapas do processo de transformação de modelos SysML para Simulink no contexto do SyMPLES.

4.3.1Extensão do SyMPLES ProfileFB

Estabelecer uma relação um-para-um entre um bloco funcional e um bloco SysML não é suficiente para especificar um sistema. Diversos parâmetros precisam ser considerados em um processo de mapeamento. Anotações podem ser necessárias para identificar que um determinado elemento criado na linguagem SysML corresponde a um determinado bloco funcional com comportamento pré-estabelecido. Essa correspondência pode ser determinada com a utilização de um conjunto de estereótipos, que podem ser aplicados a blocos SysML, com o objetivo de fornecer informação adicional para determinar uma semântica precisa.

O perfil *SyMPLES-ProfileFB* definido por Silva (2012) contém estereótipos que permitem o mapeamento entre elementos SysML e as principais classes de blocos funcionais. A Figura 21 mostra o SyMPLES-ProfileFB inicial.

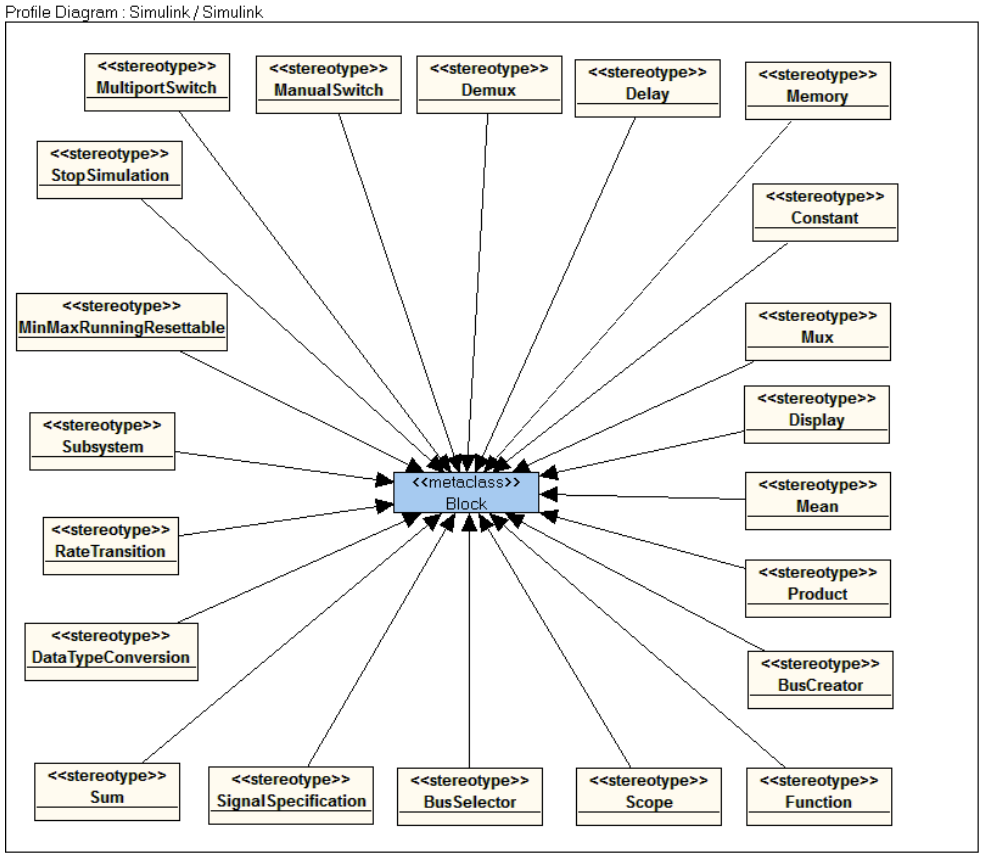

Figura 21: SyMPLES-ProfileFB inicial (Silva 2012).

Esse perfil foi estendido com a adição de mais elementos e atributos específicos a alguns estereótipos. A Figura 22 mostra o perfil estendido com os estereótipos de blocos funcionais. A primeira linha de estereótipos é abstrata e apenas organizam os diferentes tipos de estereótipos presentes na biblioteca Simulink. Esse grupo abstrato de estereótipos estendem elementos do metamodelo SysML. A maioria dos estereótipos estendem as metaclasses `Class`, `Property` e `Port`. A metaclasse `Class` é usada para estender o elemento bloco, que é representado por uma classe com estereótipo <<block>> no diagrama de definição de blocos SysML. A metaclasse `Property` estende o elemento *part* do diagrama interno de blocos. A metaclasse `Port` estende dois estereótipos que podem ser aplicados às portas de um bloco ou o elemento *part*. Eles têm como função abilitar ou desabilitar um subsistema baseado em uma verificação de valor numérico (`Enable`) ou evento (`Trigger`).

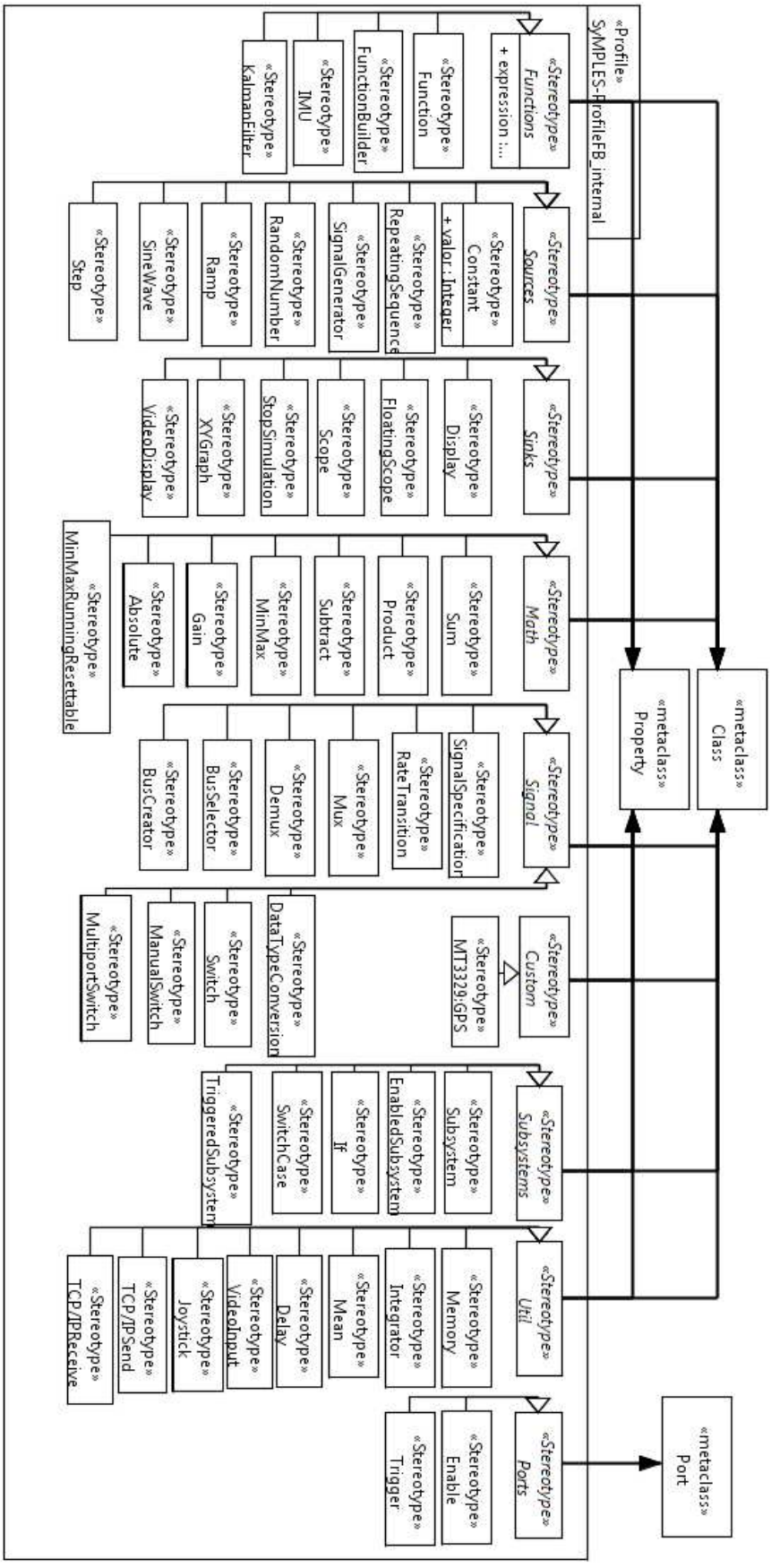

Figura 22: Representação do SyMPLES-ProfileFB estendido.

Os estereótipos agrupados pelo estereótipo abstrato `Sources` representam os estereótipos que geram sinais, enquanto o grupo `Sinks` representa estereótipos que apenas recebem sinais. O estereótipo abstrato `Custom` pode agrupar estereótipos específicos que são mapeados para blocos funcionais customizados. A plataforma Matlab/Simulink permite criar bibliotecas para representar novos blocos funcionais. Caso não exista um bloco funcional específico para um subsistema ou dispositivo mapeado, um novo estereótipo pode ser adicionado ao grupo `Custom` para ser mapeado. Por exemplo, uma nova biblioteca Simulink é criada para representar um bloco funcional de um receptor de GPS específico (modelo MT3329), assim o bloco funcional gerado na biblioteca pode ser mapeado como um modelo SysML pela adição de um estereótipo do tipo `Custom` no SyMPLES-ProfileFB. Esse estereótipo pode ser atribuído a um bloco SysML e usado para gerar o bloco funcional Simulink definido pela biblioteca específica do GPS.

Os estereótipos do SyMPLES-ProfileFB possuem atributos específicos (porém não visíveis na Figura 22). Por exemplo, estereótipos do tipo Ganho (`Gain`) e Constante (`Constant`) têm um atributo chamado `Value` (ou valor) em que um valor inicial pode ser atribuído e transformado como um parâmetro inicial para blocos funcionais Simulink. A Figura 23 mostra uma instância de um bloco SysML chamado `part1` do tipo `Batery` que representa o valor da carga de uma bateria usando o estereótipo SyMPLES-ProfileFB `Constant`. O valor inicial 10 (dez) é dado ao atributo do estereótipo e a partir disso é possível utilizar esses dados para gerar um bloco funcional do tipo `Constant` com um valor pré-definido no Simulink.

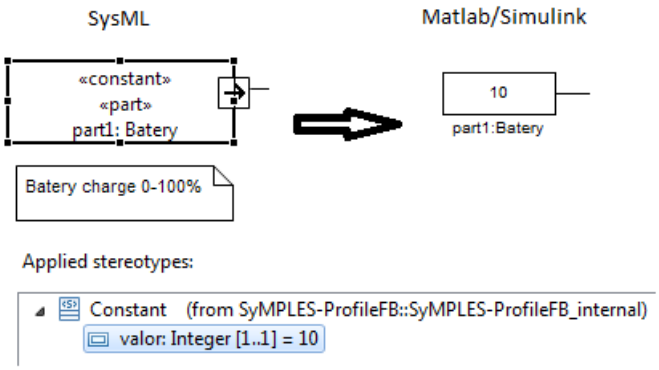

Figura 23: Exemplo de elemento SysML com um estereótipo SyMPLES-ProfileFB.

Outros estereótipos podem ter outros atributos como é o caso do `TCP/IP receive`. Atributos como `host` e `port` também podem ser mapeados para parâmetros do bloco funcional.

4.3.2 Etapas do Processo de Transformação

O processo de transformação de modelos SysML para modelos de blocos funcionais Simulink é composto por três atividades, que são:

- Gerar o modelo SysML configurado;
- Realizar a transformação ATL;
- Gerar o modelo de blocos funcionais.

A Figura 24 mostra as etapas de desenvolvimento relacionadas ao processo de transformação. O processo de transformação apenas inclui uma parte do desenvolvimento MDE de um sistema. Os modelos transformados e gerados são independentes de plataforma (ou PIM na notação MDA). O modelo Simulink final pode então ser refinado para uma plataforma específica e gerar código executável.

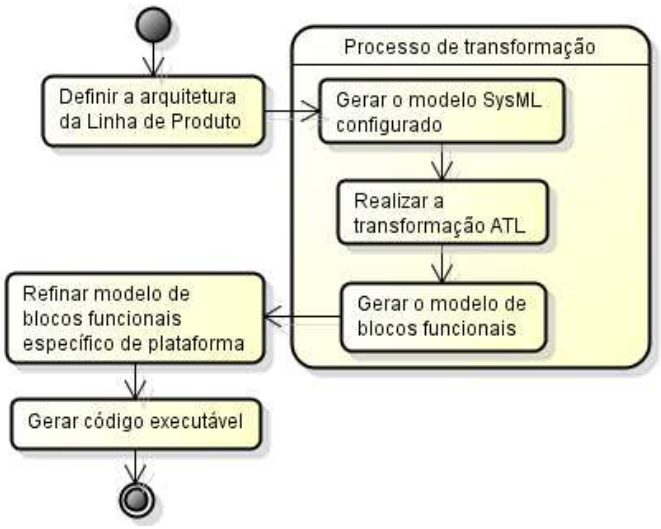

Figura 24: Atividades do processo de transformação de diagramas SysML para Simulink.

A seguir são apresentadas as três atividades do processo de transformação.

4.3.2.1 Gerar o Modelo SysML Configurado

O modelo SysML configurado é composto pelos: diagramas de definição de bloco, diagrama interno de blocos e diagramas de máquinas de estados. A Figura 25 mostra a relação dos elementos dos diagramas usados para a transformação em blocos funcionais. A raiz do modelo SysML consiste de um diagrama de definição de blocos, em que blocos com o estereótipo `subsystem` podem agrupar diagramas internos de blocos ou máquina de estados. No diagrama interno de blocos o elemento `Part` representa uma instancia de um bloco no qual também pode abrigar diagramas de máquina de estados. Apenas são mapeados os

elementos `Block` e `Part` que tem estereótipos SyMPLES-ProfileFB aplicados a eles. Portas (`Port`) podem ser usadas para representar a transição de dados entre os blocos.

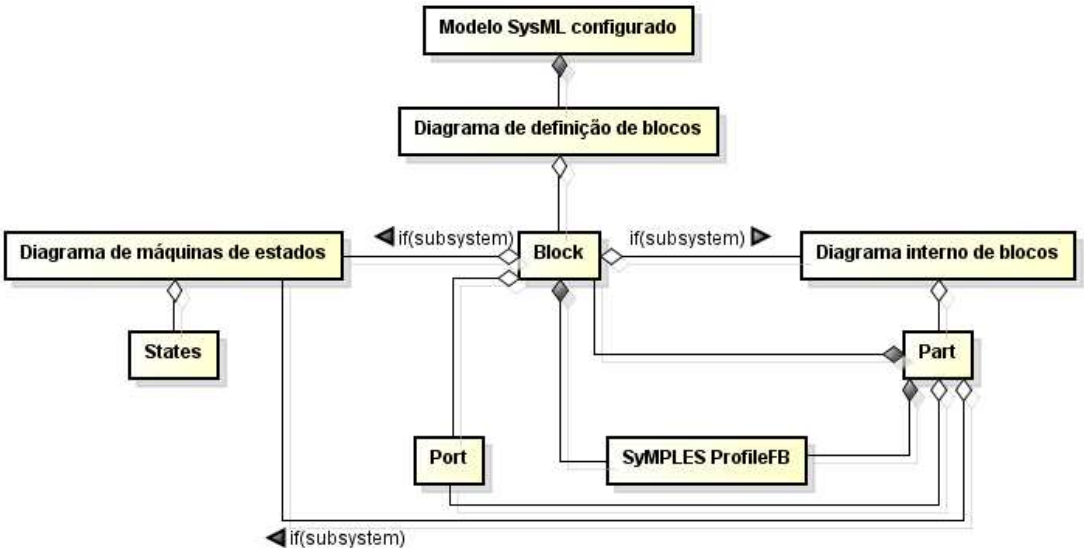

Figura 25: Relação entre os elementos dos diagramas usados na transformação.

Os elementos do diagrama de definição de blocos usados na transformação são blocos (*block*) que tem os estereótipos do SyMPLES-ProfileFB. A Figura 26 mostra um exemplo de um diagrama de definição de blocos com os estereótipos <<subsystem>> aplicados aos blocos `Sensores`, e `Sistema de navegação`. Esse exemplo é parcial e baseado na representação de um mini-VANT.

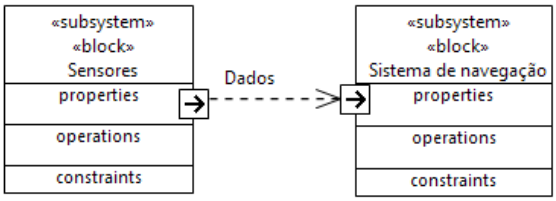

Figura 26: Exemplo de um diagrama de definição de blocos com estereótipos do SyMPLES-ProfileFB.

Os blocos podem representar um diagrama interno de blocos com elementos chamados `part`. Esse elemento é modelado como uma instância de um bloco. Por exemplo, um elemento `part` só existe se for de um tipo de bloco específico. A Figura 27 mostra um exemplo de diagrama interno de blocos com os estereótipos do SyMPLES-ProfileFB aplicados para as partes `receiver:GPS`, `part1:IMU` e `m1:Mux`.

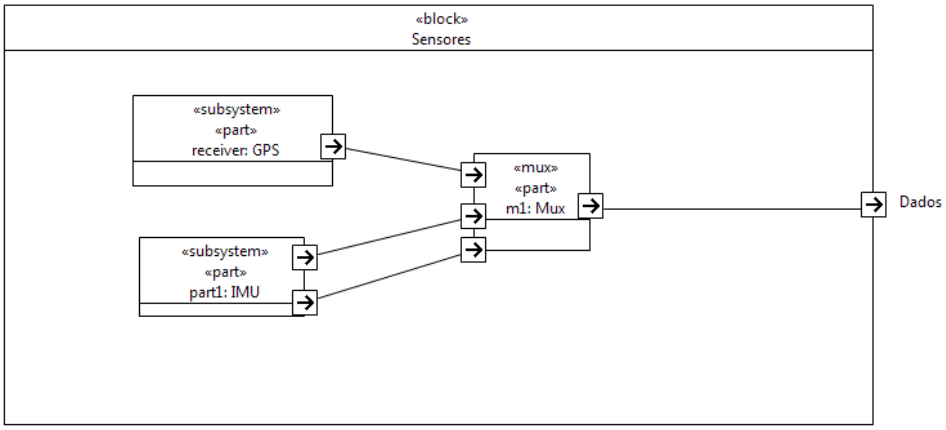

Figura 27: Exemplo de diagrama interno de blocos para o bloco Sensores com estereótipos do SyMPLES-ProfileFB.

Diagramas de máquinas de estados também são mapeados para a transformação. Um bloco com o estereótipo <<*subsystem*>> pode agrupar internamente um diagrama interno de blocos ou de máquina de estados. Um elemento `block` ou `part` pode representar um comportamento específico por uma máquina de estados. A Figura 28 mostra um exemplo de um diagrama de máquina de estados de Moore representado para o controle guia de curso do bloco `Sistema de navegação`. O diagrama de máquina de estados UML pode representar pontos de entrada (*entryPoint*) ou pontos de saída (*exitPoint*). O ponto de entrada `Sensor_data` ou o ponto de saída `Cmd` funcionam como interfaces de entrada e saída de valores para o diagrama máquina de estados. Expressões condicionais podem ser vinculadas as transições e, também, é possível atribuir valores às variáveis ao acionar determinado estado. Os elementos básicos do diagrama de máquina de estados são transformados e podem ser executados por modelos Stateflow.

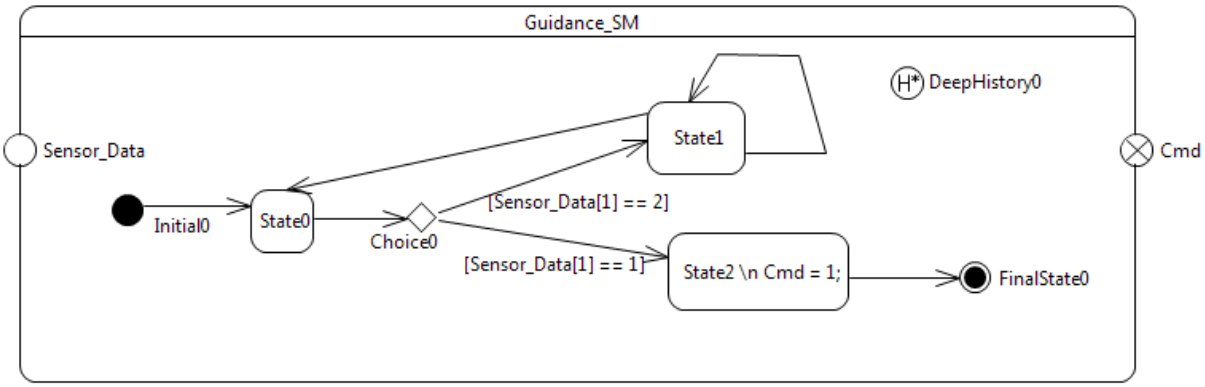

Figura 28: Exemplo de diagrama de máquina de estados para a parte “nav: Guidance”.

4.3.2.2 Realizar a Transformação ATL

A transformação ATL agrupa informações relevantes do modelo SysML e gera um arquivo XMI baseado em um metamodelo Simulink. Esse arquivo gerado então é usado para gerar o script baseado na API do Simulink na próxima atividade do processo de transformação. A Figura 29 mostra os elementos relacionados à transformação ATL em camadas de metamodelos. Os modelos UML e Notation (representação dados gráficos) são usados de entrada e como saída tem-se o modelo Simulink. Essa transformação ATL independe de ferramenta, porém, é exclusiva para editores baseados no EMF.

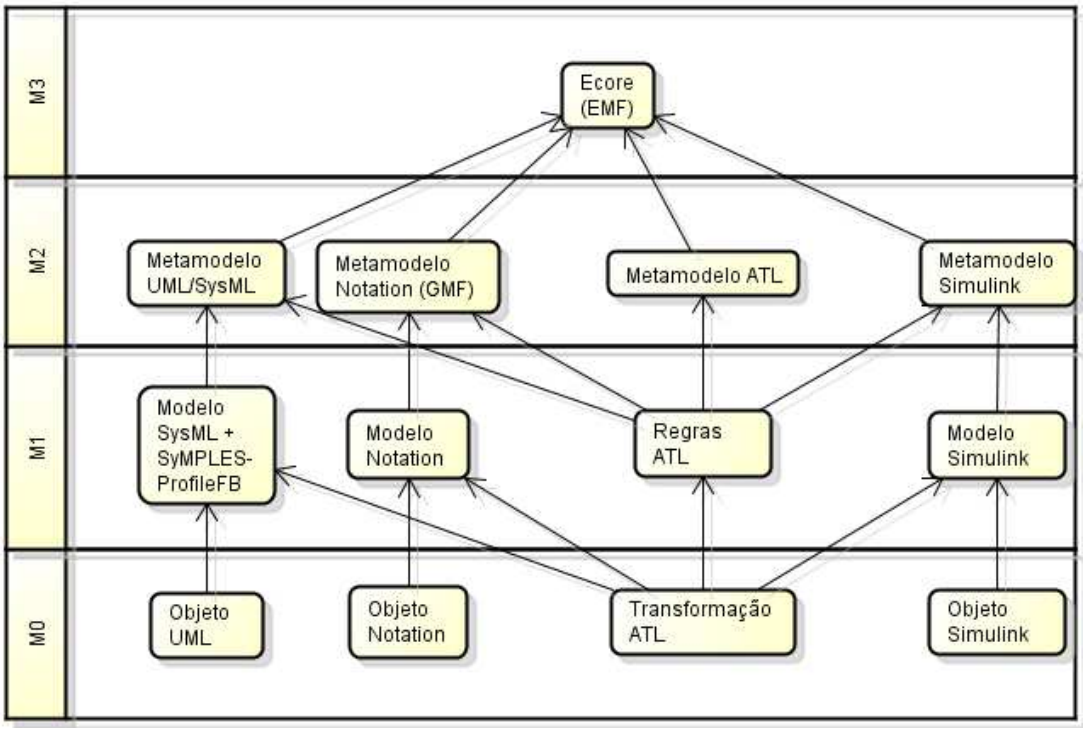

Figura 29: Elementos usados na transformação ATL em camadas de metamodelagem.

A Figura 30 mostra o conjunto de regras ATL criadas para executar a transformação (código completo no apêndice C). Nesse arquivo regras de transformação foram criadas para transformar diagramas de definição de blocos, interno de blocos e de estados. Outros diagramas UML/SysML podem ser mapeados e transformados por novas regras ATL.

```
1   -- @atlcompiler atl2006
2   -- @nsURI UML=http://www.eclipse.org/uml2/2.1.0/UML
3
4   module SysML2Simulink;
5   create OUT: Simulink from IN: NOTATION, IN2: UML;
6
7   rule Diagram2Model {
25
26  lazy rule Shape2System {
62
63  lazy rule Shape2Port {
73
74  lazy rule BasicCompartment2SystemReference {
84
85  lazy rule Shape2SystemReference {
98
99  lazy rule Edge2Line {
118
119 -- state machine rules
120 lazy rule Edge2Transition {
133
134 lazy rule Shape2SystemReference_state {
153
154 lazy rule Shape2SystemReference_state_recursive {
168
169 lazy rule Shape2SystemReference_region_states {
184
185 --helpers
186 helper context OclAny def: getPath(): String =
191
192 helper context NOTATION!Bounds def: getBounds(): String =
201
202 helper context OclAny def: getID(): String =
204
205 helper context UML!ConnectorEnd def: getConnectorLink(): String =
207
```

Figura 30: Regras ATL criadas para a transformação ATL.

O metamodelo Simulink usado para a transformação ATL é uma variação do metamodelo usado no trabalho de Biehl, DeJiu, e Törngren (2010). A Figura 31 mostra o metamodelo Simulink e no apêndice B é mostrado todo o metamodelo.

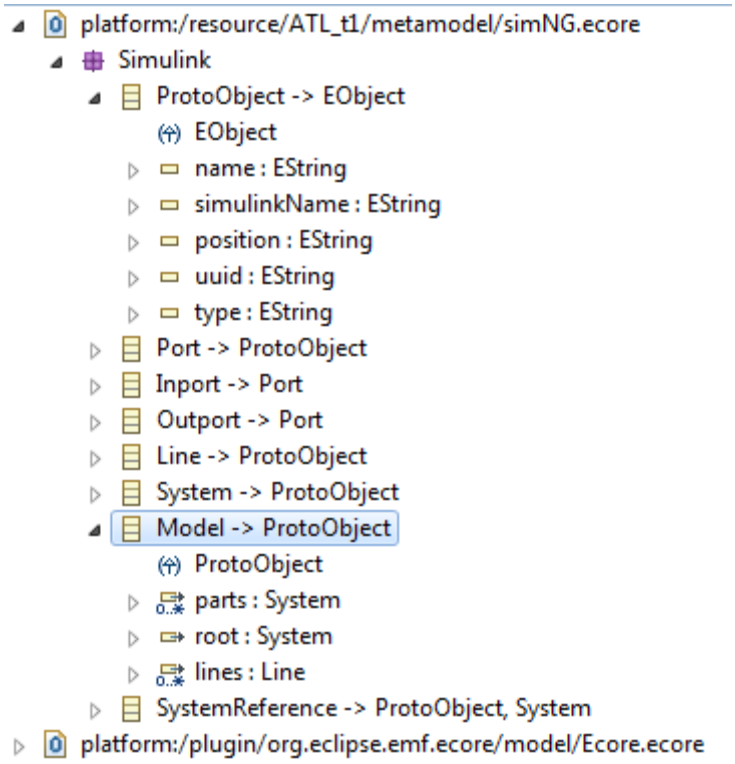

Figura 31: Representação Ecore do metamodelo Simulink.

Para o mapeamento dos diagramas de definição de blocos e internos de blocos os elementos usados são: (i) *parts* - representam blocos dos diagramas; (ii) *ports* - representam portas dos blocos ou partes de blocos; (iii) *children* – representam as partes de blocos dos diagramas internos de blocos; e (iv) *lines* – representam as conexões entre portas de ambos diagramas. Para diagramas de máquinas de estados os elementos usados são: (i) *children* – representam todos os elementos e os diferenciando usando o atributo “type”; e (ii) *lines* – representam transições entre os estados.

O metamodelo notation é usado na representação de elementos modelados para o editor gráfico SysML. A Figura 32 mostra parte do metamodelo notation.

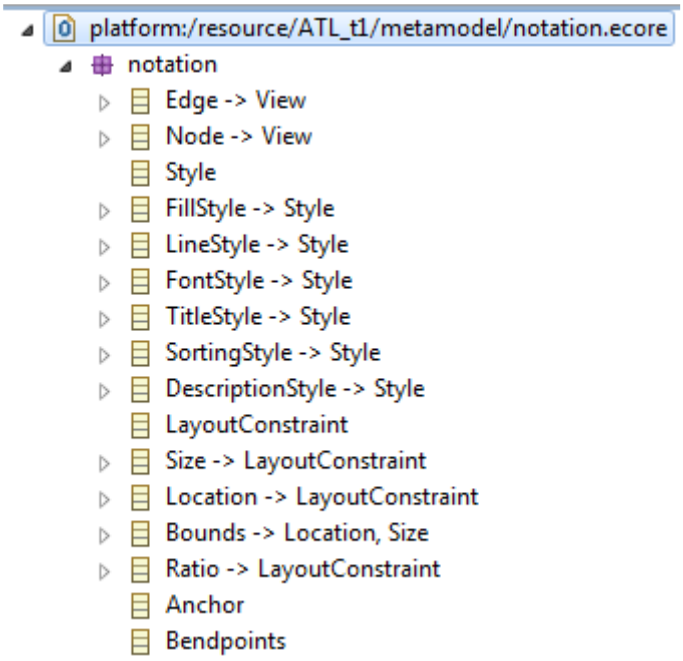

Figura 32: Representação parcial Ecore do metamodelo Notation.

Os arquivos de modelo de domínio e gráficos são sincronizados. A Figura 33 mostra um exemplo de representação XMI para um bloco chamado `Bloco3` e os artefatos gerados/relacionados (destaque no identificador xmi:id). A Figura 33a mostra a representação XMI do modelo de domínio UML da classe Bloco3. A Figura 33b mostra a representação do estereótipo <<*dataTypeConversion*>> do SyMPLES-ProfileFB para o Bloco3 no modelo de domínio UML. A Figura 33c mostra a representação do link do arquivo *notation* com o identificador do modelo de domínio UML do elemento `Bloco3` e alguns dados de tamanho (*width, height*) e posição (x,y). A Figura 33d mostra a representação XMI do elemento gerado pela transformação ATL para o `Bloco3` e os dados de posição (*position*).

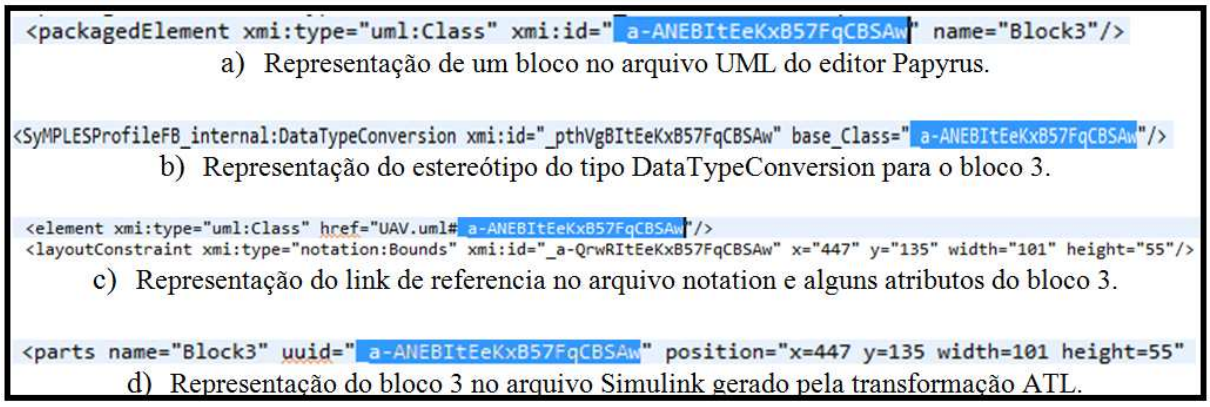

Figura 33: Representação XMI dos artefatos relacionados à transformação ATL.

Após realizar a transformação ATL o artefato gerado consiste de diagramas ordenados pela *tag* `sim:Model`. A Figura 34 mostra uma parte do arquivo gerado pela transformação ATL para os diagramas de definição de blocos e o diagrama interno de blocos dos sensores

apresentados na atividade do processo anterior para o exemplo do mini-VANT.

```
<sim:Model name="Sensor_internal" simulinkName="Sensores/SysMLmode
  <parts name="Sensores" simulinkName="_cAgogFAeEeKrzKImINUaVw" pc
    <children name="compartment_sysml_structure">
      <children name="receiver:GPS" position="x=95 y=29 width=156
        <ports name="flowport1" position="x=146 y=30 width=20 heig
      </children>
      <children name="part1:IMU" position="x=100 y=139 width=129 h
        <ports name="flowport1" position="x=119 y=6 width=20 heigh
        <ports name="flowport2" position="x=119 y=41 width=20 heig
      </children>
      <children name="m1:Mux" position="x=360 y=74 width=91 height
        <ports name="flowport1" position="x=-10 y=7 width=20 heigh
        <ports name="flowport2" position="x=-10 y=39 width=20 heig
        <ports name="flowport3" position="x=-10 y=65 width=20 heig
        <ports name="flowport4" position="x=81 y=31 width=20 heigh
      </children>
    </children>
    <lines name="connector1" type="Connector" source_uml="_GoAkcFA
    <lines name="connector2" type="Connector" source_uml="_E-JDoFA
    <lines name="connector3" type="Connector" source_uml="_GA8kcFA
    <lines name="connector4" type="Connector" source_uml="_Mbe-8FA
    <ports name="Dados" position="x=666 y=145 width=20 height=20"
  </parts>
</sim:Model>
```

Figura 34: Parte do arquivo XMI gerado pela transformação ATL.

4.3.2.3 Gerar o Modelo de Blocos Funcionais

A geração de blocos funcionais é a principal etapa do processo de transformação, responsável por criar um *script* capaz de gerar um modelo Simulink. A Figura 35 mostra o fluxo de artefatos que estão relacionados para a geração do modelo Simulink.

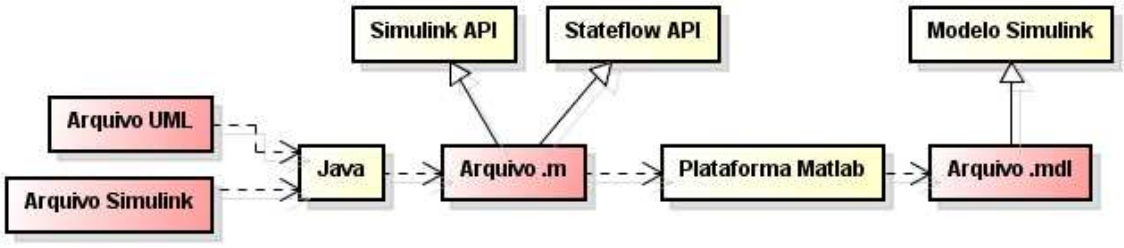

Figura 35: Fluxo de artefatos para a geração de blocos funcionais Simulink.

O arquivo Simulink gerado pela transformação ATL e o arquivo UML usado como entrada na transformação ATL são os arquivos de entrada para o código de transformação escrito em linguagem Java. O arquivo UML é necessário para obter os dados dos estereótipos SyMPLES-ProfileFB. O código Java gera um *script* usando as interfaces de aplicação (*Application Programming Interfaces - APIs*) do Simulink e Stateflow para gerar diagramas de blocos funcionais e diagrama de máquina de estados, respectivamente. Enfim, executando

o *script* na plataforma Matlab é gerado o modelo Simulink (.mdl).

Os elementos gerados para o diagrama de blocos funcionais Simulink seguem um mapeamento que é implementado em linguagem Java. O mapeamento de estereótipos SyMPLES-ProfileFB para blocos funcionais Simulink é dada pela Tabela 1. Ela consiste do tipo de estereótipo, estereótipo SyMPLES-ProfileFB, o bloco funcional Simulink mapeado e as entradas e saídas mínimas e máximas. Alguns estereótipos têm atributos relacionados representados entre parênteses que são mapeados para parâmetros específicos de um bloco funcional. Alguns blocos funcionais, por exemplo, são do tipo `built-in/Reference` que é um bloco de referencia abstrato. Para definir corretamente o tipo desse bloco é adicionado o parâmetro `SourceBlock` ao bloco de referencia com o endereço da biblioteca que se encontra o bloco funcional. Existem outros parâmetros que podem ser relacionados a expressões e valores numéricos que foram generalizados pelo Simbolo &.

Tabela 1: Mapeamento SyMPLES-ProfileFB para blocos funcionais.

ID	Tipo	SyMPLES-ProfileFB	Blocos Funcionais	Entradas	Saídas
01		Subsystem {Statemachine}	stateflow/Chart	0..*	0..*
02	Functions	Function	built-in/Fnc	1..1	1..1
03	Functions	FunctionBuilder	built-in/S-Function	1..*	1..*
04	Sources	Constant =(Value)	built-in/Constant “&”	0..0	1..1
05	Sources	RepeatingSequence (SourceBlock)	built-in/Reference “simulink/Sources/Repeating Sequence”	0..0	1..1
06	Sources	SignalGenerator	built-in/SignalGenerator	0..0	1..1
07	Sources	RandomNumber	built-in/RandomNumber	0..0	1..1
08	Sources	Ramp (SourceBlock)	built-in/Reference “simulink/Sources/Ramp”	0..0	1..1
09	Sources	SineWave	built-in/Sin	0..0	1..1
10	Sources	Step	built-in/Step	0..0	1..1
11	Sinks	Display	built-in/Display	1..1	0..0
12	Sinks	FloatingScope (Floating)	built-in/Scope “on”	1..1	0..0
13	Sinks	Scope	built-in/Scope	1..1	0..0
14	Sinks	StopSimulation	built-in/Stop	1..1	0..0
15	Sinks	XYGraph (SourceBlock)	built-in/Reference “simulink/Sinks/XY Graph”	2..2	0..0
16	Math	Sum	built-in/Sum	2..*	1..1
17	Math	Product	built-in/Product	2..*	1..1
18	Math	Subtract (Inputs)	built-in/Sum “+-#”	2..*	1..1

19	Math	MinMax	built-in/MinMax	2..*	1..1
20	Math	Gain =(Value)	built-in/Gain “&”	1..1	1..1
21	Math	Absolute	built-in/Abs	1..1	1..1
22	Math	MinMaxRunningResettable (SourceBlock)	built-in/Reference “simulink/Math Operations/MinMax Running Resettable”	2..2	1..1
23	Signal	SignalSpecification	built-in/SignalSpecification	1..1	1..1
24	Signal	RateTransition	built-in/RateTransition	1..1	1..1
25	Signal	Mux	built-in/Mux	1..*	1..1
26	Signal	Demux	built-in/Demux	1..1	1..*
27	Signal	BusCreator	built-in/BusCreator	1..1	1..*
28	Signal	BusSelector	built-in/BusSelector	1..*	1..1
29	Signal	Switch =(Threshould)	built-in/Switch “&”	3..3	1..1
30	Signal	ManualSwitch (SourceBlock)	built-in/Reference “simulink/Signal Routing/Manual Switch”	2..2	1..2
31	Signal	MultiportSwitch	built-in/MultiportSwitch	1..*	1..1
32	Signal	DataTypeConversion	built-in/DataTypeConversion	1..1	1..1
33	Subsystems	Subsystem	built-in/Subsystem	0..*	0..*
34	Subsystems	EnabledSubsystem	built-in/Subsystem built-in/EnablePort	0..*	0..*
35	Subsystems	TriggeredSubsystem	built-in/Subsystem built-in/TriggerPort	0..*	0..*
36	Subsystems	If =(Expression)	built-in/If “&”	1..*	2..2
37	Subsystems	SwitchCase	built-in/SwitchCase	1..1	1..*
38	Util	Memory	built-in/Memory	1..1	1..1
39	Util	Integrator	built-in/Integrator	1..1	1..1
40	Util	Mean (SourceBlock)	built-in/Reference “dspstat3/Mean”	1..1	1..1
41	Util	Delay	built-in/Delay	1..1	1..1

Apesar da Tabela 1 apresentar o mapeamento entre estereótipos SyMPLES-ProfileFB e blocos funcionais Simulink, existem algumas restrições que devem ser consideradas para uma transformação correta. Por exemplo, diagramas de definição de blocos e diagramas internos de blocos não podem ter portas que recebem mais que uma ligação. Também existem elementos não mapeados como portas do tipo `inout`. A Figura 36 mostra uma parte do *script* gerado para o modelo SysML usado como exemplo nas atividades anteriores do processo.

```
function mini_1(modelname)
 if nargin == 0
    modelname = 'mini_1';
    Rc6fkgFAeEeKrzKImINUaVw = 'mini_1/Sensores';
    RjxnNUFAeEeKrzKImINUaVw = 'mini_1/Sistema de navegação';
    RYxM0FAeEeKrzKImINUaVw = 'mini_1/Sensores/receiver:GPS';
    RlKmgFAeEeKrzKImINUaVw = 'mini_1/Sensores/part1:IMU';
    RIOJb0FAfEeKrzKImINUaVw = 'mini_1/Sensores/m1:Mux';
    Rtas8FAfEeKrzKImINUaVw = 'mini_1/Sistema de navegação/nav:Guidance';

end
open_system(new_system(modelname));
add_block('built-in/SubSystem', Rc6fkgFAeEeKrzKImINUaVw);
set_param(Rc6fkgFAeEeKrzKImINUaVw, 'Position', [400 330 525 454]);
RfRpCEFAeEeKrzKImINUaVw = strcat(Rc6fkgFAeEeKrzKImINUaVw, '/Dados_0');
add_block('built-in/Outport', RfRpCEFAeEeKrzKImINUaVw);

add_block('built-in/SubSystem', RjxnNUFAeEeKrzKImINUaVw);
set_param(RjxnNUFAeEeKrzKImINUaVw, 'Position', [631 330 761 456]);
Rn5TbcFAeEeKrzKImINUaVw = strcat(RjxnNUFAeEeKrzKImINUaVw, '/Sensor_Data_0');
add_block('built-in/Inport', Rn5TbcFAeEeKrzKImINUaVw);

add_line('mini_1', 'Sensores/1', 'Sistema de navegação/1','autorouting','on');
add_block('built-in/SubSystem', RYxM0FAeEeKrzKImINUaVw);
set_param(RYxM0FAeEeKrzKImINUaVw, 'Position', [95 29 251 100]);
RGoAkcFAfEeKrzKImINUaVw = strcat(RYxM0FAeEeKrzKImINUaVw, '/flowport1_0');
add_block('built-in/Outport', RGoAkcFAfEeKrzKImINUaVw);
```

Figura 36: Trecho do script Simulink gerado para o exemplo do mini-VANT.

Os elementos gerados para o diagrama de máquina de estados Stateflow também seguem um mapeamento implementado em linguagem Java. A Tabela 2 mostra os elementos mapeados que consistem de elementos UML do diagrama de estados, o elemento stateflow mapeado, as propriedades relacionadas e valores atribuidos. Valores `self` são valores não fixos obtidos pelo diagrama que são atribuídos as propriedades.

Tabela 2: Mapeamento do diagrama de estados UML para diagrama de estados Simulink.

ID	UML	Stateflow	Property	Value
0	State	State	Name Position	Self Self
1	Region	State	Name Position Decomposition IsGrouped	Self Self PARALLEL_AND true
2	Initial	Transition	SourceEndPoint Destination DestinationOClock	[xsourceysource] ID Direction
3	FinalState	State	Name Position Debug.Breakpoints.onExit	Self Self true
4	ShallowHistor	Junction	Position.Radius	Radius

	y		Position.Center Type	Center HISTORY
5	DeepHistory	Junction	Position.Radius Position.Center Type	Radius Center HISTORY
6	Fork	Junction	Position.Radius Position.Center	Radius Center
7	Join	Junction	Position.Radius Position.Center	Radius Center
8	Choice	Junction	Position.Radius Position.Center	Radius Center
9	Junction	Junction	Position.Radius Position.Center	Radius Center
10	Terminate	State	Name Position Debug.Breakpoints.onExit	Self Self true
11	Transition	Transition	Source Destination SourceOClock DestinationOClock	ID ID Direction Direction
12	EntryPoint	Input	Name	Self
13	ExitPoint	Output	Name	Self

A Figura 37 mostra uma parte do script gerado do diagrama de máquina de estados do exemplo do mini-VANT.

```
sfnew('state_machine');
rt = sfroot;
m = rt.find('-isa', 'Simulink.BlockDiagram', 'Name', 'state_machine');
ch = m.find('-isa','Stateflow.Chart');

sm_1 = Stateflow.Data(ch(1));
sm_1.set('Scope', 'Input');
sm_1.set('Name', 'Sensor_Data');
sm_1 = Stateflow.Data(ch(1));
sm_1.set('Scope', 'Output');
sm_1.set('Name', 'Cmd');
sm_Rr4t8FAfEeKrzKImINUaVw = Stateflow.State(ch(1));
sm_str = sprintf('Region1');
sm_Rr4t8FAfEeKrzKImINUaVw.LabelString = sm_str;
sm_Rr4t8FAfEeKrzKImINUaVw.Position = [31 30 695 236];
sm_RMNr0FAfEeKrzKImINUaVw = Stateflow.State(ch(1));
sm_str = sprintf('State0');
sm_RMNr0FAfEeKrzKImINUaVw.LabelString = sm_str;
sm_RMNr0FAfEeKrzKImINUaVw.Position = [175 121 40 40];
sm_RAi7nIFAgEeKrzKImINUaVw = Stateflow.State(ch(1));
sm_str = sprintf('State1');
sm_RAi7nIFAgEeKrzKImINUaVw.LabelString = sm_str;
sm_RAi7nIFAgEeKrzKImINUaVw.Position = [424 66 61 46];
sm_RA1TJgFAgEeKrzKImINUaVw = Stateflow.State(ch(1));
sm_str = sprintf('State2 \n Cmd = 1;');
```

Figura 37: Trecho do script gerado para o Stateflow do exemplo do mini-VANT.

4.4 Considerações Finais

Este capítulo apresentou um processo de transformação que estende a abordagem SyMPLES, definido neste trabalho de mestrado, em que foram usadas técnicas MDE para realizar a transformação de modelos SysML para blocos funcionais Simulink. No SyMPLES sistemas embarcados são representados por blocos em SysML com variabilidades de uma PL. A extensão SyMPLES-ProfileFB visa facilitar a especificação de alto nível de sistemas embarcados com a definição de um conjunto de estereótipos. Os estereótipos são mapeados entre um diagrama de blocos ou diagrama interno do bloco em SysML com as principais classes da abordagem de blocos funcionais. Com a resolução das variabilidades é criado um produto da PL que pode ser transformado em blocos funcionais Simulink. Esse modelo Simulink gerado pode ser detalhado para facilitar a geração de código.

Capítulo

5

Avaliação da Transformação de Modelos na Engenharia de Aplicação

5.1. Considerações Iniciais

Este capítulo apresenta uma avaliação do processo de transformação de modelos na engenharia de aplicação da abordagem SyMPLES, em que são discutidos os benefícios e as limitações em relação a outros trabalhos. Inicialmente são apresentados as dinâmicas de movimento de aeronaves e o sistema controlador de vôo usado na avaliação para contextualizar o subsistema utilizado como domínio do exemplo de aplicação. Na sequência, é apresentado um exemplo de aplicação do processo de transformação, em que foram criados modelos SysML da placa controladora do piloto automático Yapa 2 do VANT Paparazzi (YAPA 2011). Com base no subsistema de controle de movimento de rolagem do sistema controlador de vôo do VANT, foram construídos os modelos interno de blocos e representado uma máquina de estados hipotética para tratar comandos gerados pelo piloto automático. O exemplo elaborado mostra os artefatos gerados pelo processo de transformação e uma simulação para o modelo Simulink gerado. Finalmente, são apresentados os parâmetros utilizados em uma avaliação comparativa.

5.2 Dinâmicas de Movimento

A estabilidade e controle de uma aeronave são mais difíceis de manter que um carro ou navio, pois, uma aeronave é estabilizada em três eixos de rotação. Em uma aeronave a mudança em um tipo de estabilidade afeta os outros dois. O modelo de VANT usado na avaliação é um tipo de aeronave, assim as dinâmicas de movimento existentes para aeronaves se aplicam também a esse tipo de VANT.

Os atuadores básicos de uma aeronave são chamados *ailerons*, *elevators* (ou estabilizador) e *rudder* (leme). Esses atuadores são chamados servos e representam as superfícies de controle da aeronave. Os servos permitem a execução do movimento de uma aeronave.

Baseado no conhecimento de dinâmica de vôo de aeronaves, comandos enviados os servos que controlam os *ailerons* causam o movimento chamado *roll*, enquanto comandos enviados para o *rudder* causam o movimento chamado *yaw* (vide Figura 38). Comandos enviados para o *elevator* causam movimentos de *pitch* na aeronave (FAA 2008).

Primary Control Surface	Airplane Movement	Axes of Rotation	Type of Stability
Aileron	Roll	Longitudinal	Lateral
Elevator/ Stabilator	Pitch	Lateral	Longitudinal
Rudder	Yaw	Vertical	Directional

Figura 38: Controle, movimento, rotação e estabilidade de uma aeronave (FAA 2008).

Para executar um movimento de *roll* é necessário ajustar os ailerons de forma inversa um em relação ao outro. Por exemplo, a Figura 39 mostra a execução de um comando de *roll* à esquerda, que significa ajustar o aileron da direita para baixo (levantar asa) e o aileron da esquerda para cima (abaixar asa) (FAA 2008).

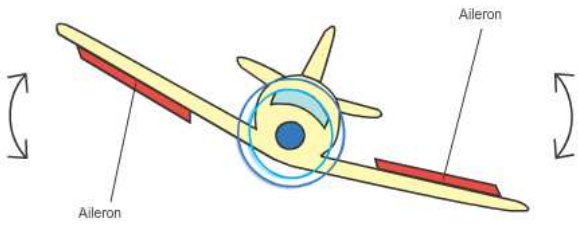

Figura 39: Movimento de roll de uma aeronave.

O atuador *rudder* permite o movimento de *yaw* que ajuda o movimento de *roll* (porém em outro eixo de rotação). Os movimentos de *roll* e *yaw* são combinados e fazem com que a aeronave realize o movimento de rolagem (ou *roll-yaw*), pois os controles combinados do *aileron* e *rudder* melhoram o desempenho de vôo. Porém, as entradas de controle para os ailerons causam uma resposta de rolagem maior comparado ao controle de entrada do *rudder* (FAA 2008). A Figura 40 mostra o movimento de yaw de uma aeronave.

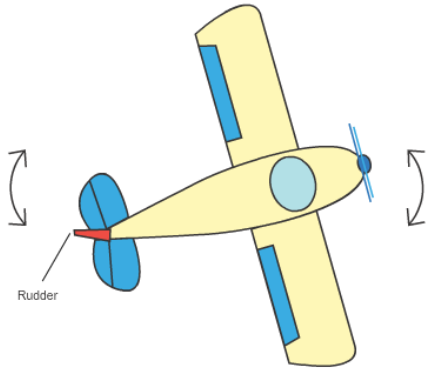

Figura 40: Movimento de yaw de uma aeronave.

Para executar um movimento de *pitch* os dois *elevators* agem como um só fazendo a frente do VANT subir (empinar) ou descer. Por exemplo, ajustar o *elevator* para cima faz com que o VANT suba (vide Figura 41), enquanto ajustar para baixo faz o VANT descer.

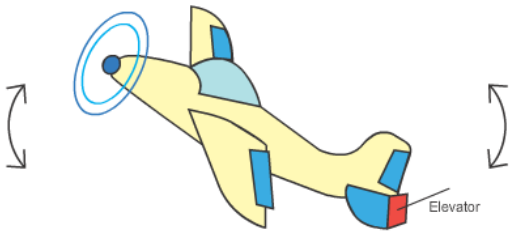

Figura 41: Movimento de pitch de uma aeronave.

Comandos de movimento gerados pelo controle de navegação do VANT são tratados no sistema controlador de vôo. Por exemplo, o controle de vôo recebe um comando de *roll* de X graus. O controle de vôo então gera os comandos apropriados para os servos a fim de executar o movimento desejado.

5.3 Sistema de Controle de Vôo

Basicamente um VANT é pilotado automaticamente por um computador embarcado chamado

Sistema de Controle de Vôo (*Flight Control System - FCS*) (Pratt 2000). Os sensores de vôo (acelerômetros, giroscópios, GPS, sensores de pressão atmosférica) e o piloto automático são seus principais componentes. Esse sistema lê informações dos sensores e guia o VANT pelo seu plano pré-determinado (piloto automático). Em um típico vôo cruzeiro, o VANT opera a condição de vôo específica e alcança os pontos de navegação (*waypoints*) por meio de comandos de rolagem (*roll*), em que a velocidade (*airspeed*) e altitude são fixas.

Um sistema de piloto automático consiste de um software (`autopilot software`) que executa no VANT (vide Figura 42). Esse software é encarregado de guiar o VANT (`UAV`) usando dados gerados pelos sensores e, também, repassando eles a uma estação terrestre (`ground control station`). O software do piloto automático inicialmente é carregado com um plano de vôo (`flight plan`) que guia o VANT por uma sequência de pontos de coordenadas (*waypoints*). Baseado na posição atual do VANT os controles de navegação (`navigation control loop`) e altitude gerenciam as superfícies de controle (`plane control surfaces`) da aeronave (ou atuadores) que o guia para o próximo *waypoint* do plano de vôo.

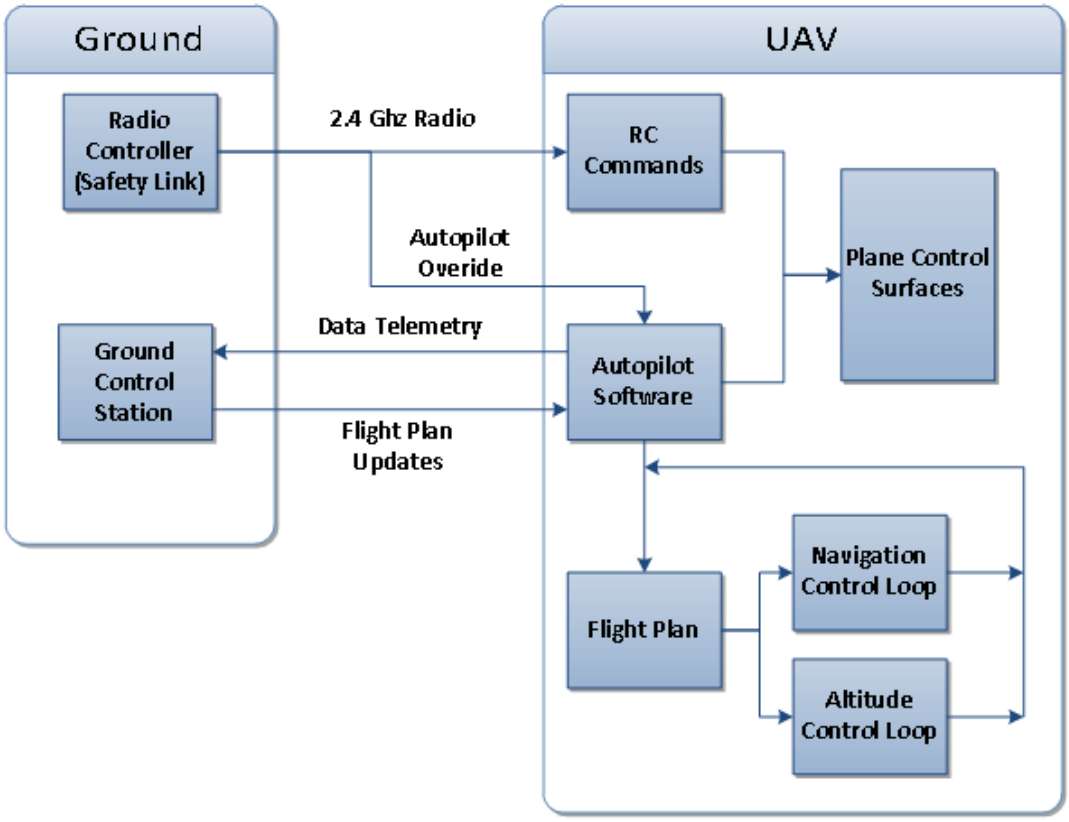

Figura 42: Fluxo de dados entre as interfaces de um sistema de piloto automático e a estação terrestre (Coleman et al., 2012).

A estação terrestre (`Ground`) geralmente é um computador portátil que executa um software de planejamento que recebe dados dos sensores do VANT e os apresenta por uma interface gráfica. A estação terrestre fornece ao usuário a possibilidade de alterar o plano de vôo do VANT em tempo real caso seja necessário. Como segurança o VANT tem uma interface que permite mudar o modo de vôo e enviar controles manuais (`RC Commands`) via radio (*`radio controller`*).

Um dos objetivos do controlador de vôo é tratar comandos gerados pelo sistema de navegação. A Figura 43 mostra sua estrutura do controle de vôo para o VANT Paparazzi que é dividida em dois principais controles, um para controle vertical (altitude) e outro horizontal (navegação), em que as saídas geram os controles finais para os servos. O controle de *roll (`roll loop`)* também trata comandos de *yaw*.

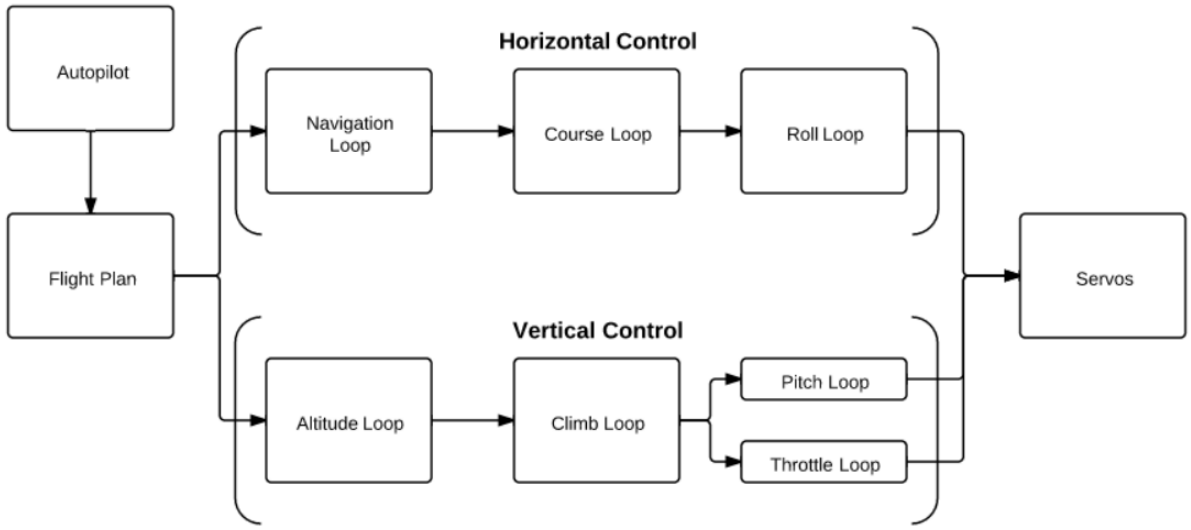

Figura 43: Estrutura do controle de vôo Paparazzi (Coleman et al., 2012).

Comandos de *roll* são gerados pelo controlador de orientação de *waypoints* durante o vôo. Por exemplo, o controle de *roll* recebe um comando de *`roll 10`* (graus) e então gera os comandos apropriados para os servos. A síntese dos sinais de orientação ou waypoints não é incluída no escopo deste trabalho, e é assumido que esses sinais geram comandos de *roll*.

Sistemas controladores de vôo consistem de subsistemas primários e secundários. Controles de comandos de *roll, pitch e yaw* consistem os subsistemas primários. Controles de comandos para os atuadores como *flaps, rudder trim* e *spoilers*, por exemplo, são considerados subsistemas secundários que melhoram o desempenho da aeronave ou aliviam o piloto das forças de controle excessivas (FAA 2008).

5.4 Projeto Paparazzi

Paparazzi é um projeto livre e de código aberto para hardware e software, destinado a criar versáteis sistemas de piloto automático para aeronaves de asa fixa ou multicópteros. A característica única do piloto automático Paparazzi é a capacidade de combinar medições inerciais e/ou sensores infravermelhos (*infrared thermopiles*) para definir a altura. Assim, é fornecida uma robusta e precisa estimativa de altura, que não requer calibração terrestre e pode se recuperar a partir de qualquer altitude (Enac 2003).

A Figura 44 mostra a placa do piloto automático Yapa 2 do projeto Paparazzi. A placa controladora é uma versão adaptada da placa Arduíno (Arduino 2012). O dispositivo de comunicação via radio usado é o XBee, que é ligado diretamente á placa controladora. Existem outros projetos como o Ardupilot (Drones 2012) que também usam placas modificadas baseadas no arduíno.

Figura 44: Placa do piloto automático Paparazzi Yapa 2.

O sistema de piloto automático Yapa 2 gerencia o controle autônomo e a estabilidade de vôo. O sistema de vôo tem três modos de vôo:

- **Modo manual:** controla diretamente os atuadores via radio;
- **Modo auto1:** controla a estabilidade permitindo o VANT voar em linha reta;
- **Modo auto2:** controla de forma autônoma o VANT que segue um plano de vôo que contém blocos de vôo pre configurados.

Para executar o piloto automático é necessário que todos os componentes estejam

configurados corretamente. O software do piloto automático pode compilar nos microprocessadores das famílias ARM7 serie LPC21xx e STM32 usando cinco arquivos de configuração XML (vide Figura 45), que são compilados juntos e então carregados na placa controladora Yapa 2 (Coleman et al. 2012). Os arquivos são:

- `Estrutura (Airframe)`: o arquivo de configuração da aeronave agrupa configurações específicas dos subsistemas e sensores. As superfícies de controle (servos) e seus comportamentos também são incluídos em uma seção no arquivo de configuração. Isso deixa o sistema flexível com a adição de novos sensores ou subsistemas;
- `Telemetria (Telemetry)`: o arquivo de configuração de telemetria define as mensagens usadas pela comunicação da estação terrestre. As mensagens transmitidas são mostradas na estação terrestre em tempo real e são transmitidas em um período de tempo. Essas mensagens são salvas para depuração e análise após o vôo;
- `Plano de vôo (Flight Plan)`: o arquivo de plano de vôo contém os pontos (*waypoints*) em que o VANT deve navegar em relação à coordenada inicial do GPS.
- `Radio`: o arquivo de configuração do radio interage com a interface do controlador de radio. Pilotos automáticos de VANTs precisam ter um link de comunicação extra para controlar manualmente o VANT caso haja alguma falha. Essa característica é uma regulação requerida da Administração Aeronáutica Federal (*Federal Aviation Administration - FAA*) para VANTs civis.
- `Configurações (Settings)`: o arquivo de configuração de definições permite alterar valores em tempo de vôo para diversos subsistemas. Os valores dos parâmetros são alterados pelo software na estação terrestre e quando salvos são enviados ao VANT.

Figura 45: Arquivos de configuração piloto automático Paparazzi Yapa 2 (Coleman et al. 2012).

As placas usadas no projeto Paparazzi permitem a configuração de diferentes combinações de sensores. Novos controles podem ser ajustados ou implementados. A vantagem do projeto Paparazzi é que ele permite o usuário criar subsistemas customizados para suportar sensores que não são compatíveis para o piloto automático (*autopilot*) (Enac 2003).

5.5 Aplicação do Processo de Transformação

Nessa seção é apresentado um exemplo de aplicação para o processo de transformação apresentado no Capítulo 4, em que foi criado uma arquitetura de PL inicial para o Yapa 2 usando modelos SysML. Um modelo SysML é gerado pela configuração da PL e transformado em um modelo Simulink.

5.5.1 Arquitetura Inicial da PL para Yapa 2

Um dos objetivos de um controlador de vôo de um VANT é tratar comandos de navegação e gerar comandos específicos para os atuadores. Um exemplo de controlador de vôo para o VANT Paparazzi do tipo aeronave com asa fixa é apresentado nessa seção. O controle de vôo criado considera que o VANT esteja em um vôo de cruzeiro em que ele já se encontra no ar e pronto para ativar o piloto automático. Foram representados alguns elementos particulares já existentes e relacionadas com a placa do piloto automático Yapa 2. O controle de vôo é usado

para adicionar variabilidades a fim de gerar uma arquitetura de PL para a configuração de modelos SysML seguindo a abordagem SyMPLES.

A Figura 46 mostra um exemplo de uma possível arquitetura inicial para o VANT Paparazzi. A arquitetura é representada por um diagrama de definição de blocos SysML. Os blocos com estereótipos `<<subsystem>>` representam futuros subsistemas no Matlab/Simulink e possuem diagramas internos. Estereótipos do tipo `<<variationPoint>>` representam pontos de variação da PL incluída. Linhas pontilhadas direcionadas e conectadas por portas simbolizam a passagem de sinais entre os subsistemas.

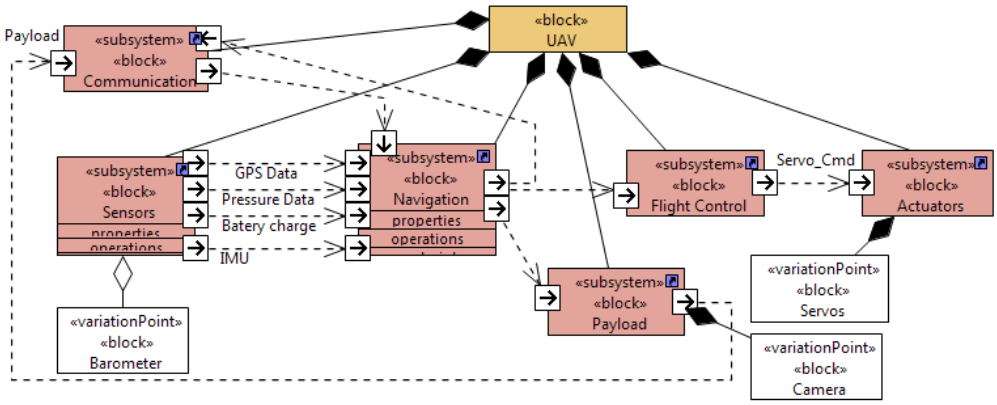

Figura 46: Arquitetura inicial para o VANT (UAV) Paparazzi.

O subsistema de sensores (*`Sensors`*) é representado por um diagrama interno de blocos SysML e apresenta três tipos de sensores que podem ser usados na construção de VANTs (vide Figura 47). O receptor GPS do modelo MT3329 pode ser usado no projeto Paparazzi. Para barômetros (*`Barometer`*) que medem a pressão atmosférica dois modelos podem ser usados: BMP085 e MS5611. O sensor do tipo termopilha pode ser usado em vez de barômetros tradicionais. Como existem dois modelos de barômetros eles são exclusivos, ou seja, apenas um deles realmente pode ser usado em um VANT real. Assim, o estereótipo `<<alternative_XOR>>` foi atribuído aos dois elementos para serem resolvidos por uma configuração específica. Uma vez atribuído o estereótipo de variabilidade a um elemento é necessário criar um comentário com estereótipo `<<variability>>` e preencher os atributos relacionados a ele dizendo quais são os elementos envolvidos e ponto de variação para a resolução da variabilidade.

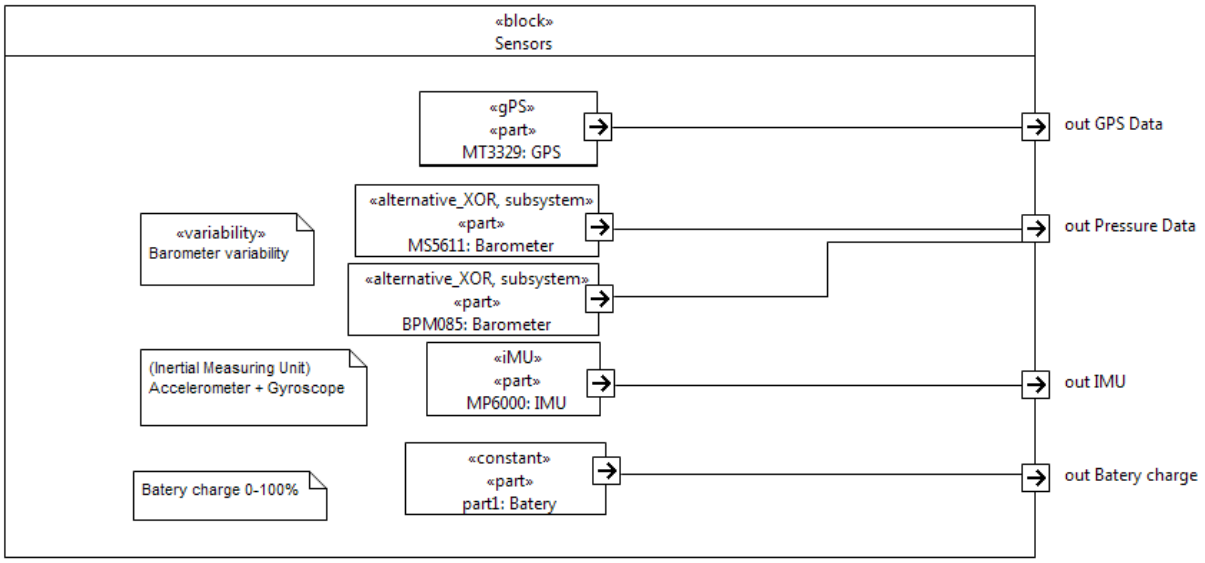

Figura 47: Diagrama interno de blocos para o subsistema Sensors.

O bloco `MT3329:GPS` representa um receptor GPS que é aplicado o estereótipo customizado do SyMPLES-ProfileFB chamado `GPS:MT3329` apresentado visualmente como <<gPS>>. Este estereótipo é mapeado a um bloco funcional de uma biblioteca Simulink customizada criada especificamente para representar esse modelo do receptor GPS. Outras bibliotecas Simulink podem ser criadas para representar outros blocos funcionais e mapeados ao SyMPLES-ProfileFB.

O controle de navegação ilustrado pela Figura 48 mostra o bloco do piloto automático que recebe como entrada dados dos sensores e comandos recebidos pelo sistema de comunicação (telemetria). A partir desses dados são gerados comandos agrupados para o sistema de controle de vôo (*`Navigation Cmds`*) e dependendo do ponto da trajetória ele pode ativar um dispositivo de carga (*payload*) pela saída *`PayloadActivator`*.

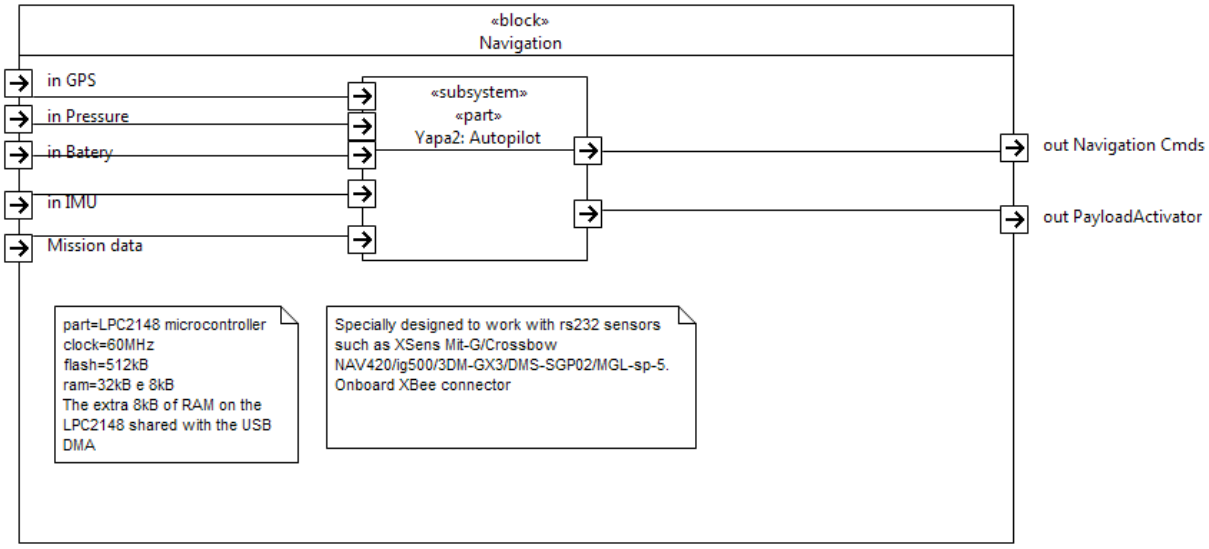

Figura 48: Diagrama interno de blocos para o subsistema Navigation.

A Figura 49 mostra a representação interna do subsistema de carga (*Payload*). Existe uma variabilidade a ser resolvida pela PL relacionada à câmera a ser utilizada. Uma câmera RGB ou infravermelha são exclusivas e apenas uma delas deve permanecer no produto final. A imagem da câmera pode ser enviada à estação terrestre se o sinal *payload_activator* habilitar a porta com estereótipo *<<enable>>* do SyMPLES-ProfileFB.

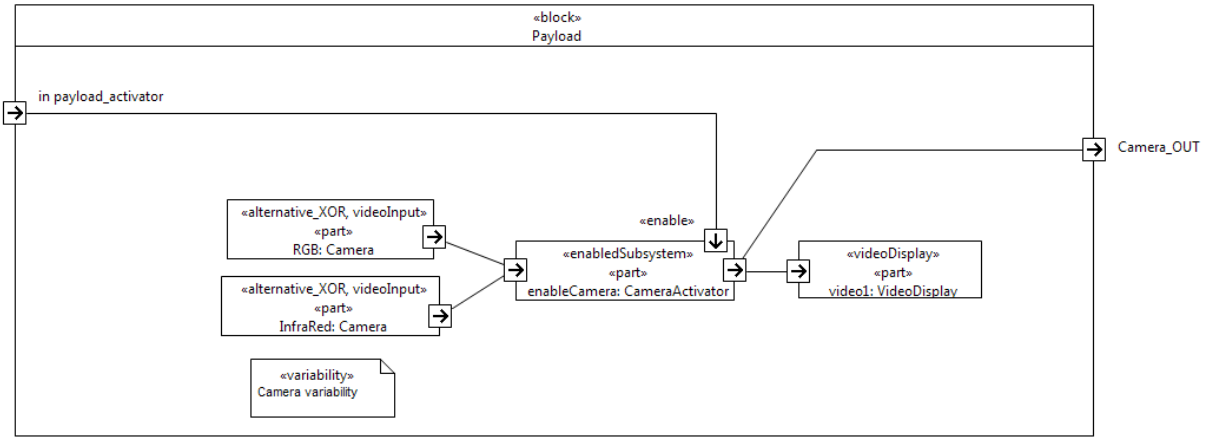

Figura 49: Diagrama interno de blocos para o subsistema Payload.

A Figura 50 mostra o controle de vôo (*Flight Control*) composto pelos controladores de movimento horizontais (*roll controller*) e verticais (*pitch and throttle controllers*). Cada um desses controladores possui o estereótipo <<subsystem>>, assim o elemento parte chamado *loop* do tipo de bloco *Roll Controller* tem um diagrama interno de blocos. Esse diagrama interno de blocos está

associado ao bloco *Roll Controller*. Assim, qualquer elemento part desse tipo de bloco tem o mesmo diagrama interno de blocos.

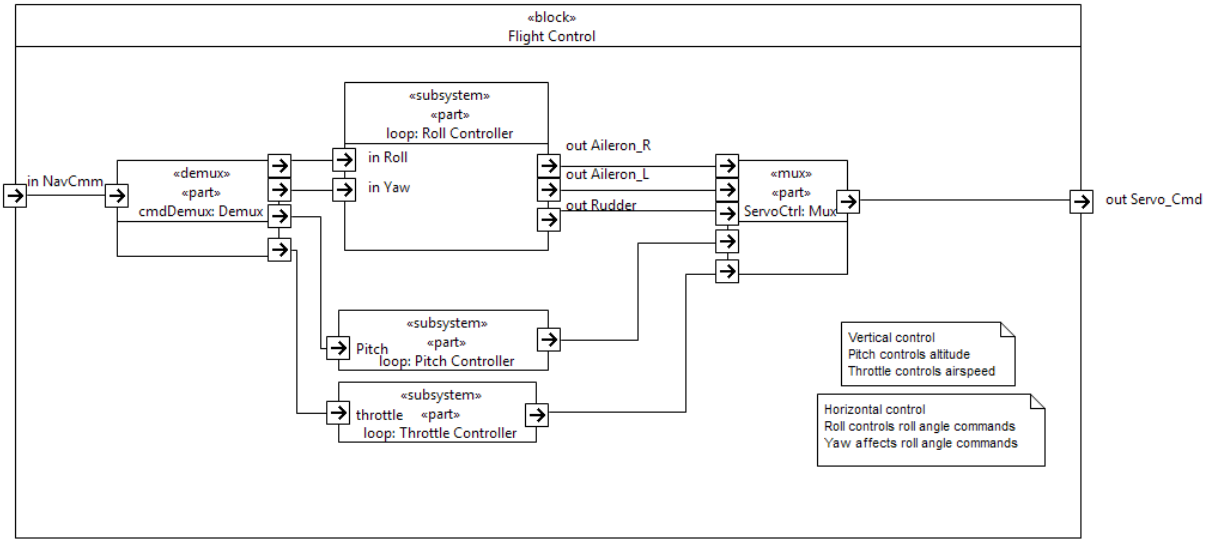

Figura 50: Diagrama interno de blocos para o subsistema Flight Control.

A Figura 51 mostra o diagrama interno ao elemento parte *loop:Roll Controller* que além de um diagrama interno de blocos também é possível representar diagramas de máquinas de estados. Nesse exemplo foi criada uma máquina de estados que define valores para os atuadores de acordo com o comando de *Roll* ou *Yaw* recebido pelo subsistema de navegação. É assumida uma faixa de valores fixo que varia de -45 a +45 (graus) para os comandos de entrada e saída (servos). Essa faixa de valores foi usada para simplificar o exemplo, pois os valores reais da faixa de valores são outras e são distintas entre entrada e saída. Os estados são nomeados com expressões em que o símbolo \n representa uma quebra de linha que é usada para diferenciar o nome do estado e a atribuição de valores no modelo Stateflow. As transições são representadas por expressões condicionais em que valores de entrada e os valores locais dos servos são verificados. Se a condição foi alcançada um estado é ativado e é atribuído valores de ajuste para os comandos dos servos.

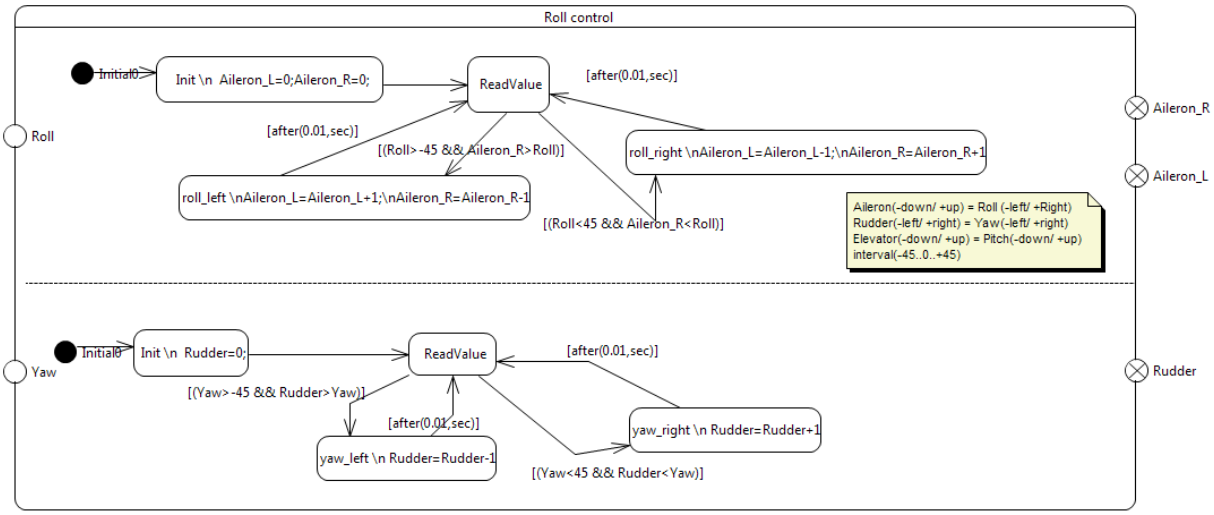

Figura 51: Diagrama de máquina de estados para o subsistema Roll Controller.

A Figura 52 mostra a representação dos atuadores do VANT em que é aplicado o estereótipo `<<scope>>`. Esse estereótipo é mapeado para Simulink e funciona como um visualizador de dados. Variabilidades nos servos também podem ser adicionadas, e de acordo com as dinâmicas de movimento de uma aeronave os *ailerons* possuem uma maior influencia sobre o movimento de *roll* que o *rudder*. Assim, um VANT pode ter apenas *ailerons* para tratar movimento de *roll.* Portanto, para este exemplo foi atribuída à variabilidade opcional `<<optional>>` no atuador `rudder`.

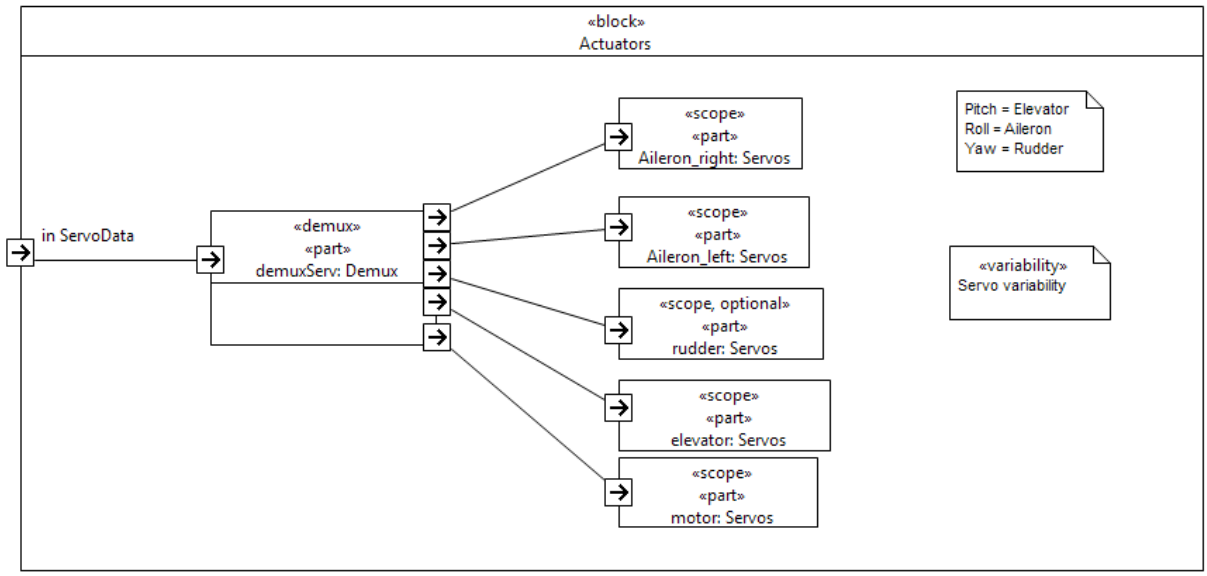

Figura 52: Diagrama interno de blocos para o subsistema *Actuators*.

5.5.2 Transformação do Modelo Yapa 2

Para realizar a transformação do modelo SysML para Simulink de acordo com o processo de transformação apresentado no capítulo 4, inicialmente é necessário criar o modelo SysML configurado para um produto específico. A atividade 1 do processo de transformação permite a geração desse modelo. Variabilidades relacionadas ao barômetro, tipo de câmera e atuador rudder opcional foram escolhidas e o modelo SysML configurado foi gerado. Após obter o modelo SyMPLES configurado, a atividade 2 do processo de transformação requer uma transformação intermediária ATL, necessária para a geração do modelo de blocos funcionais. O apêndice D mostra o arquivo gerado pela transformação ATL para o controlador de vôo Yapa 2. Esse arquivo é usado para gerar o script Matlab pela atividade 3 do processo de transformação. No apêndice A é apresentado o script gerado pelo processo de transformação para o controle de vôo.

5.5.3 Controle de Vôo Simulink Gerado

O controlador de vôo para o Simulink foi gerado a partir da execução do script Matlab gerado pela atividade 3 do processo de transformação apresentado no Capítulo 4. Essa seção mostra os subsistemas transformados para o exemplo do sistema de controle de vôo do Yapa 2.

A Figura 53 mostra a arquitetura inicial transformada para Simulink. Apenas os elementos com estereótipos SyMPLES-ProfileFB são considerados na transformação. Os conectores são transformados apenas se os subsistemas que enviam e recebem dados possuem portas que estão conectadas.

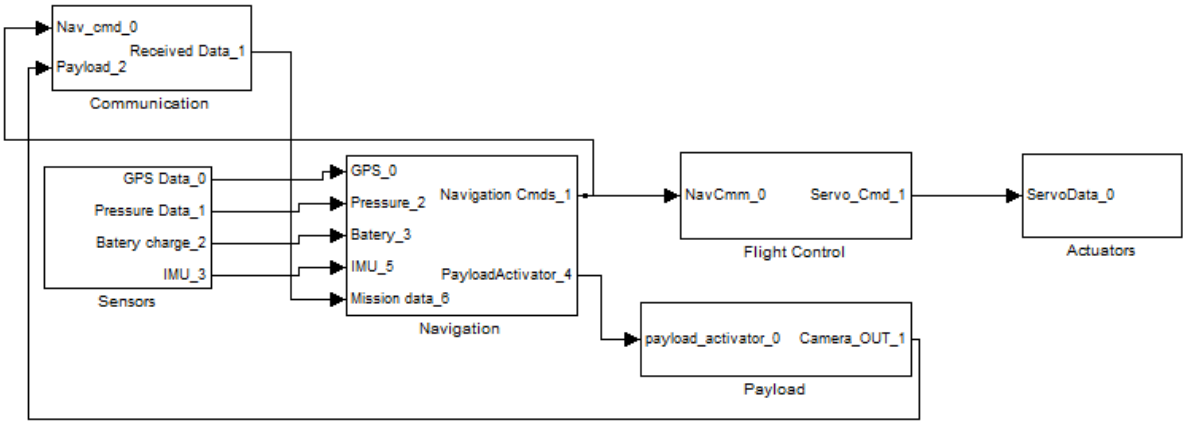

Figura 53: Arquitetura inicial para o VANT Paparazzi transformado para Simulink.

A Figura 54 mostra a transformação para o modelo Simulink do subsistema de sensores. O elemento `MP6000:IMU` em SysML tem o estereótipo `<<IMU>>` e este foi

mapeado para um bloco funcional já existente da biblioteca Simulink, enquanto o elemento `MT3329:GPS` foi mapeado a um bloco funcional criado em uma biblioteca Simulink que representa esse dispositivo. Caso não exista um estereótipo específico mapeado pelo SyMPLES-ProfileFB pode-se usar o estereótipo `<<subsystem>>` para representar o bloco de forma genérica e no Simulink especificá-lo.

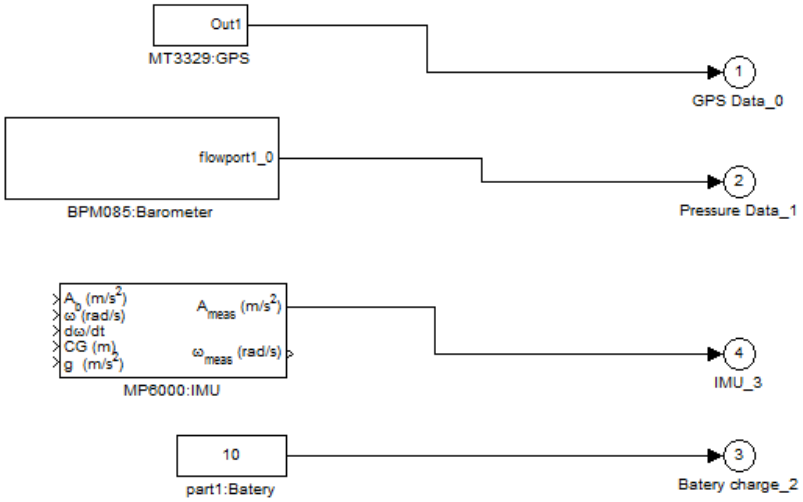

Figura 54: Subsistema de sensores transformado para o Simulink.

A Figura 55 mostra o subsistema de navegação transformado como um subsistema genérico. No Simulink esse bloco pode ser desenvolvido, porém não faz parte do escopo deste trabalho.

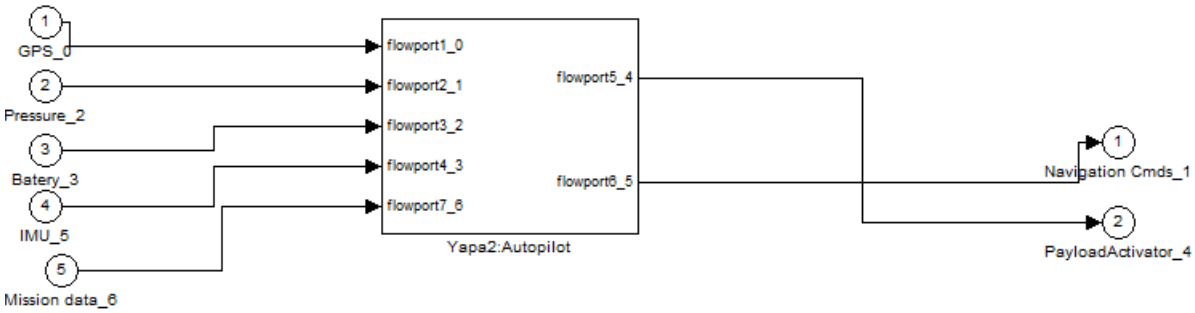

Figura 55: Subsistema de navegação transformado para o Simulink.

A Figura 56 mostra o subsistema de carga (*payload*) gerado para o Simulink. Nesse exemplo o sinal `payload_activator` pode ativar (sinal de valor 1) ou desabilitar (sinal valor 0) a imagem da câmera.

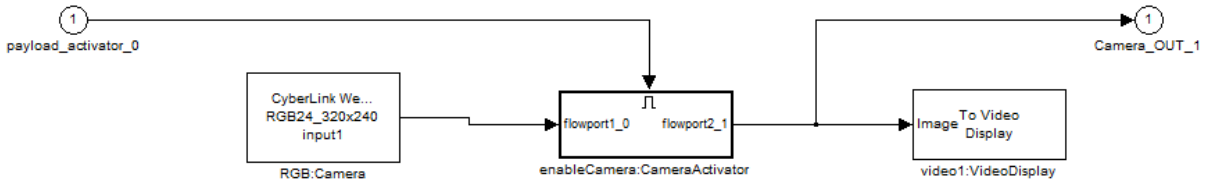

Figura 56: Subsistema de payload transformado para o Simulink.

A Figura 57 mostra o subsistema de controle de vôo gerado para o Simulink. Nesse exemplo o bloco *`loop: Roll Controller`* em SysML é relacionado com o modelo Stateflow chamado *`Roll control`*. Inicialmente logo após a transformação o elemento *`Roll control`* é mantido dentro do subsistema *`loop: Roll Controller`*, mas para facilitar a apresentação ele foi movido para fora de seu subsistema substituindo-o.

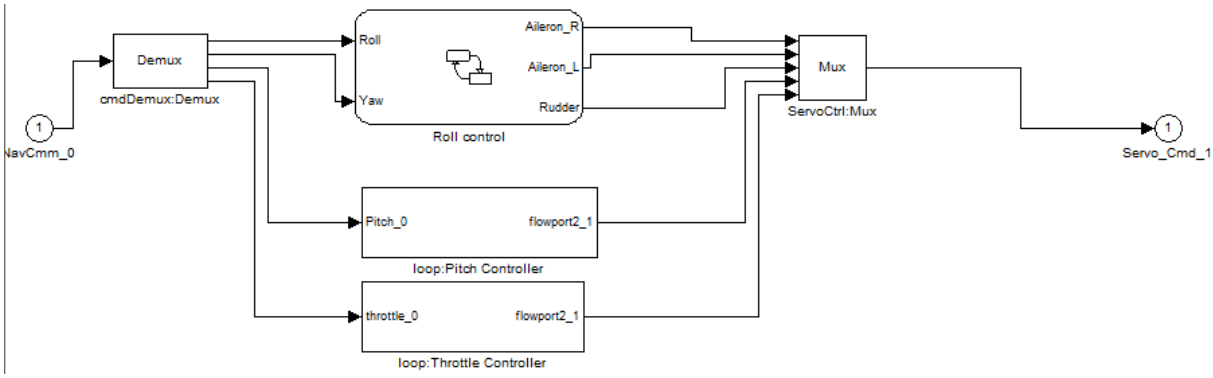

Figura 57: Subsistema de controle de vôo transformado para o Simulink.

A Figura 58 mostra o diagrama de máquina de estados SysML transformado para Stateflow. Nesse exemplo o nome do estado é definido pela primeira linha e as subsequentes são atribuições aos valores de ajuste para os comandos dos servos.

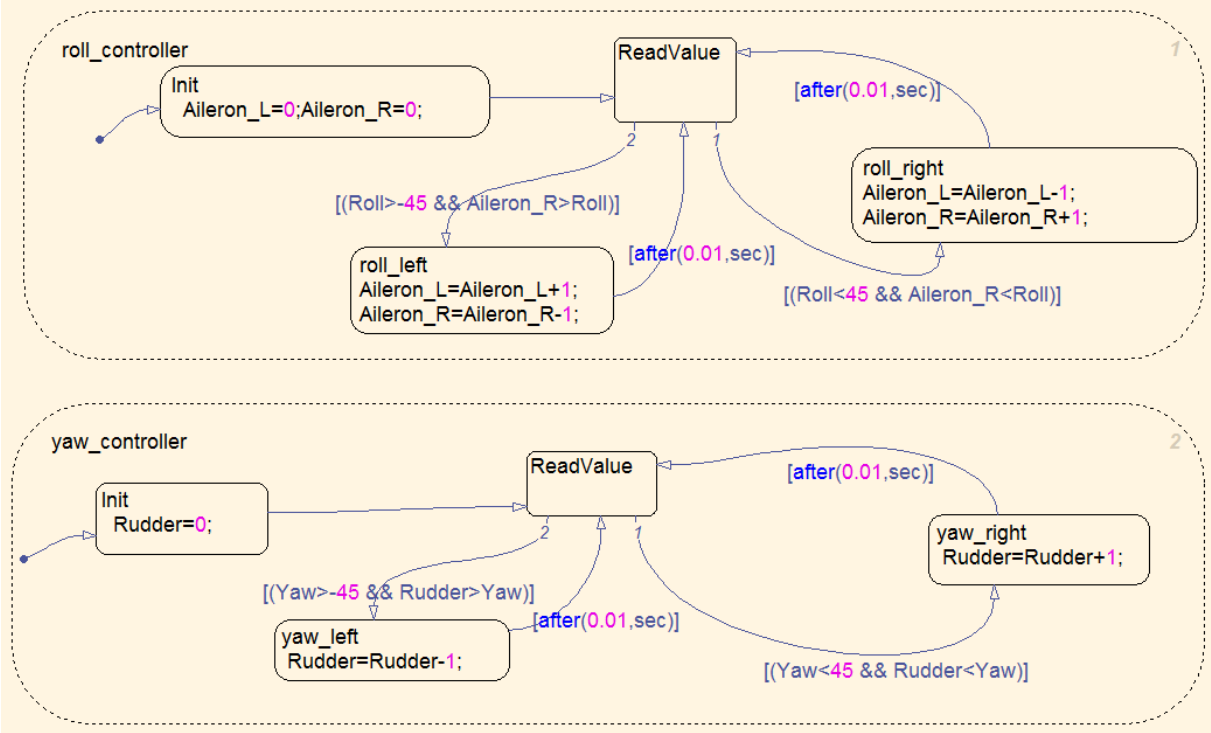

Figura 58: Diagrama stateflow gerado pela máquina de estados UML.

5.6 Simulação do Controle de vôo Gerado

No modelo Simulink do controle vôo gerado foram adicionados elementos extras para permitir a simulação do controle de *roll* usado como exemplo para o VANT Paparazzi. A Figura 59 mostra blocos funcionais adicionados no subsistema de comunicação que são usados para simular comandos manuais recebidos pela estação terrestre.

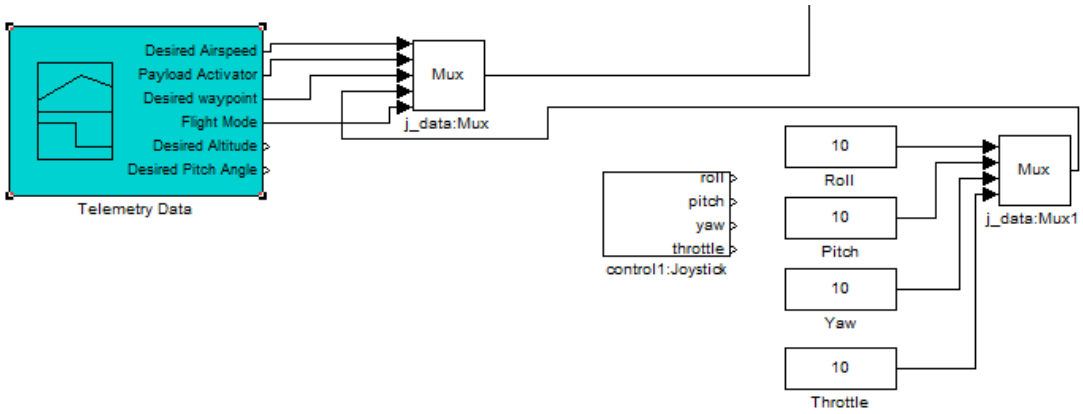

Figura 59: Exemplo de dados recebidos pelo subsistema de comunicação.

Os sinais gerados são agrupados por um bloco do tipo *Mux* e repassados ao controle de navegação pela porta `Received Data_1`. O elemento `control1:Joystick` pode ser

usado para capturar sinais de um *joystick* real e agrupados pelo bloco `j_data:Mux` para servir como sinais no modo do tipo de vôo *`manual`*, porém, foi atribuído o valor fixo 10 como exemplo para a simulação.

Casos de teste podem ser adicionados com blocos funcionais Simulink do tipo gerador de sinais (*`signal builder`*). O bloco chamado *`Telemetry Data`* da Figura 59 permite gerar sinais de diferentes tipos e com tempo específico como mostra a Figura 60.

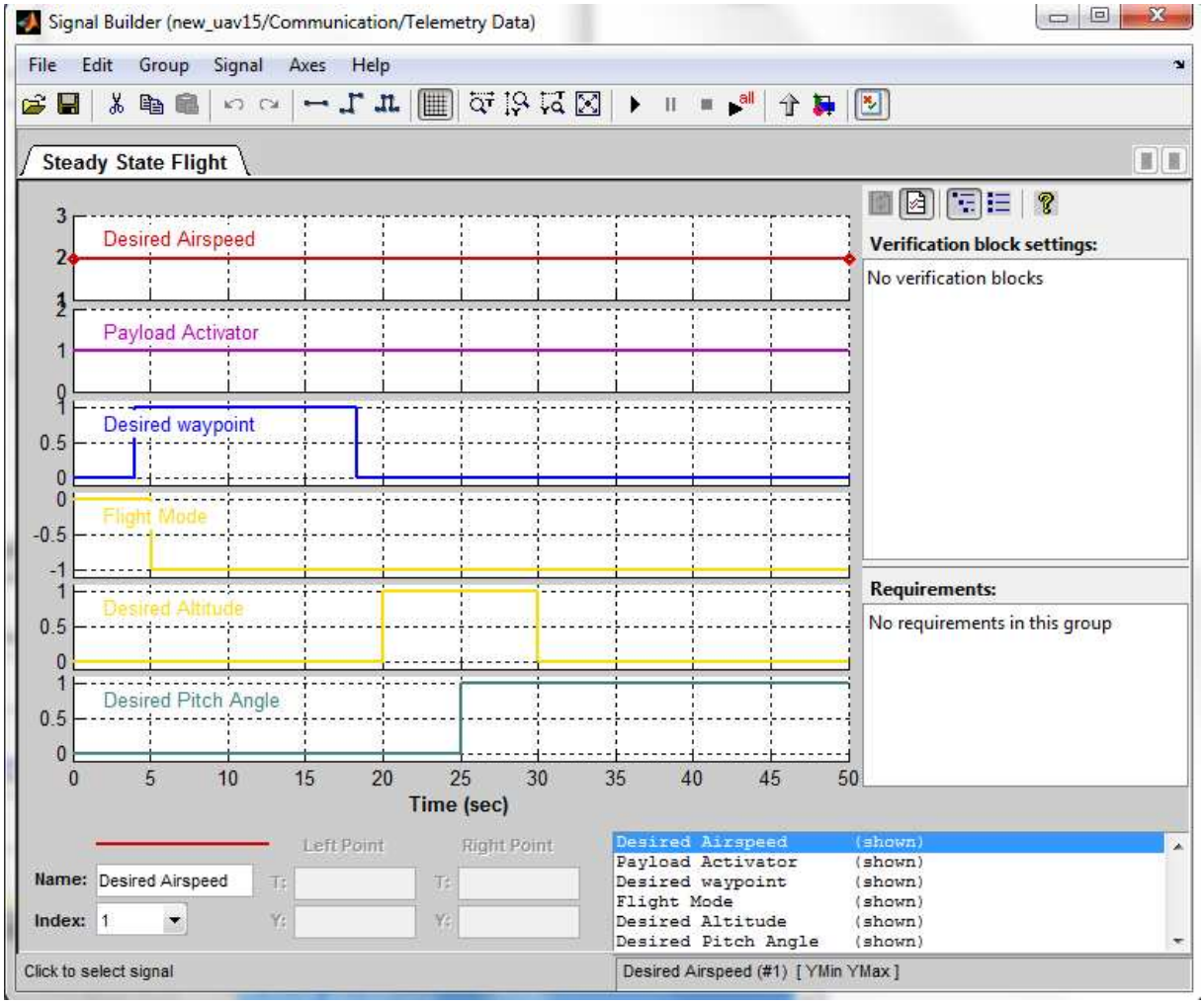

Figura 60: Exemplo de dados gerados para a simulação.

A Figura 61 mostra o bloco funcional `Autopilot Controls` do tipo `Signal Builder` foi criado para simular possíveis sinais gerados pelo piloto automático como comandos de `Roll`, `Yaw`, `Pitch` e `Throttle`. Observe na Figura 60 que o `Flight Mode` permanece em valor 0 até os primeiros 5 segundos, depois disso torna-se o valor -1. Esse valor é verificado pelo elemento `Switch` da Figura 61 o que significa que os 5

primeiros segundos o piloto automático está desativado e repassando os comandos manuais com valor 10.

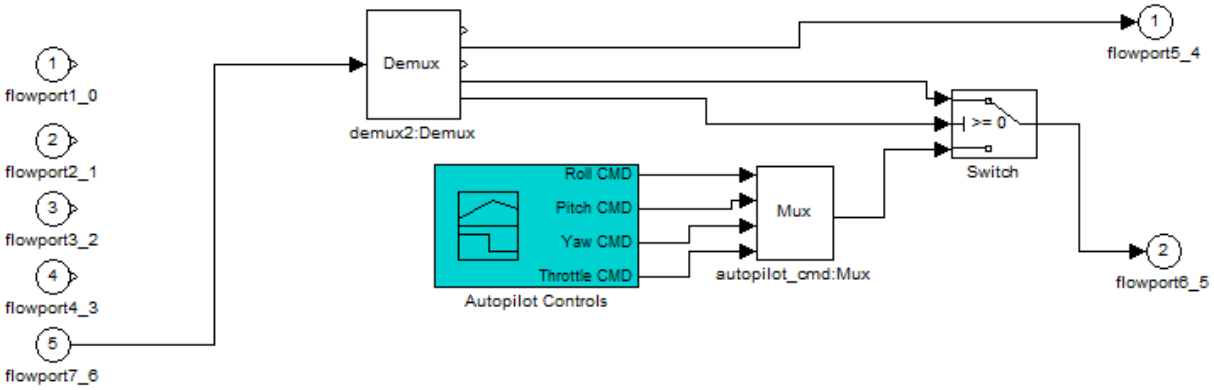

Figura 61: Exemplo de comandos para o piloto automático.

A Figura 62 mostra um exemplo de comandos (casos de teste) criados pelo gerador de sinais. Os valores variam no intervalo de -45 a +45 (supõe-se a faixa como o ângulo do movimento). No segundo 10 o piloto automático envia um sinal de comando "`Roll 45`" e "`Pitch 45`". O comando de *roll* é atualizado gradualmente e é diminuído até o valor -45 no segundo 35. Em seguida é gerado o comando "`Roll 45`". O comportamento desse sinal é simular o deslizar de um *joystick*. Considere que o *joystick* gera comandos de *`Roll`* no eixo horizontal (eixo x) e comando de *`Pitch`* no eixo vertical (eixo y). No segundo 10 o *joystick* é deslizado totalmente à direita e, gradualmente, é retornado totalmente à esquerda até o segundo 35. Quando atingido o segundo 35 o *joystick* é deslizado totalmente à direita novamente. O comando de *pitch* é gerado junto aos outros comandos. Assim, até o segundo 10 o *joystick* é deslizado para trás até o segundo 10 e gradualmente desliza o manche ao centro novamente.

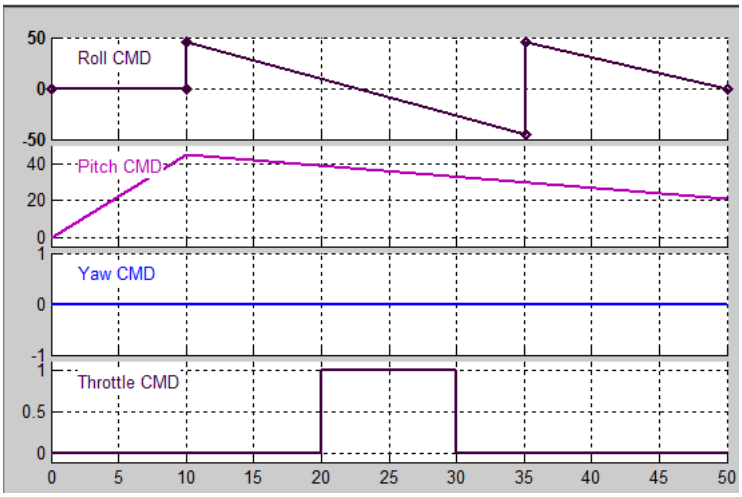

Figura 62: Exemplo de comandos do piloto automático gerados para a simulação.

A Figura 63 mostra a máquina de estados stateflow gerada para o controle de rolagem em execução. Comandos de `Roll` e `Yaw` são lidos e tratados para gerar valores aos servos.

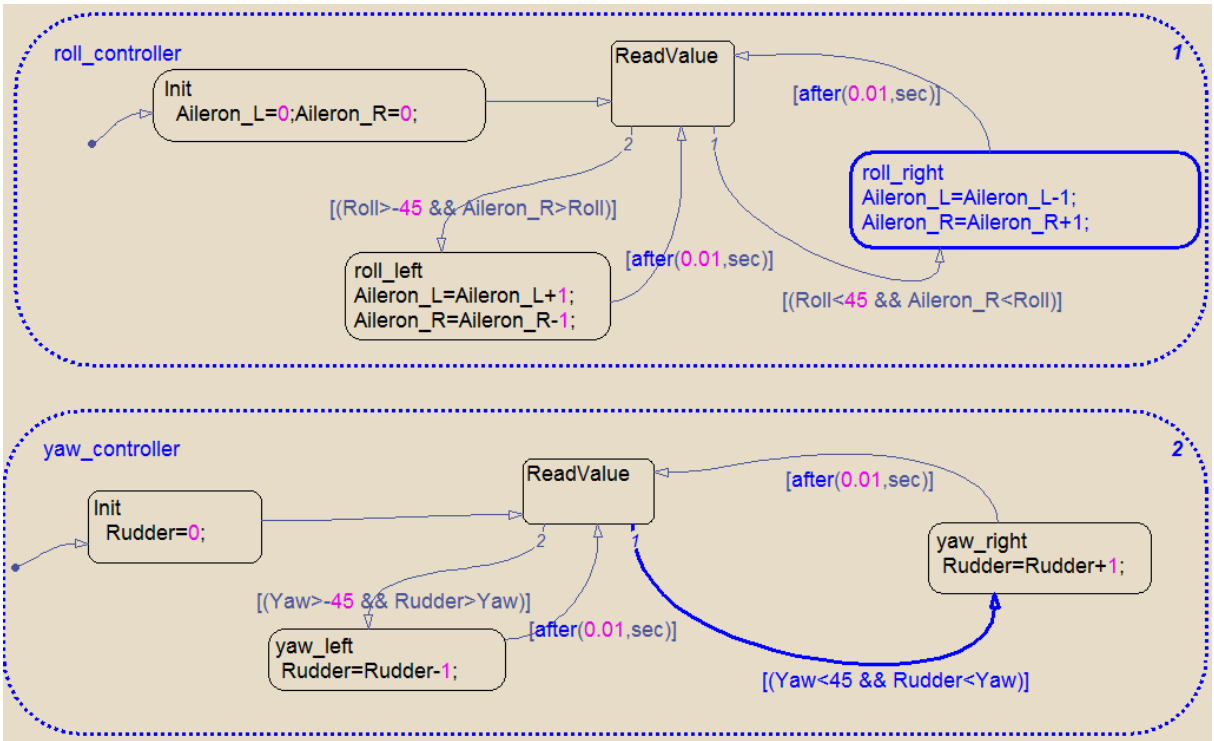

Figura 63: Diagrama stateflow em execução do controle de roll.

A Figura 64 mostra visualmente o resultado da simulação para os servos `Aileron_right` (direito) e `Aileron_left` (esquerdo). Até o segundo 5 foram enviados comandos de "`Roll 10`". Em seguida os sinais da Figura 62 foram enviados ao controlador de roll que ajustaram os valores dos servos para executar o movimento. No segundo 10 com o

comando `Roll 45` o VANT deve executar o *roll* à direita e para tal o *aileron* da direita precisa subir enquanto o da esquerda descer.

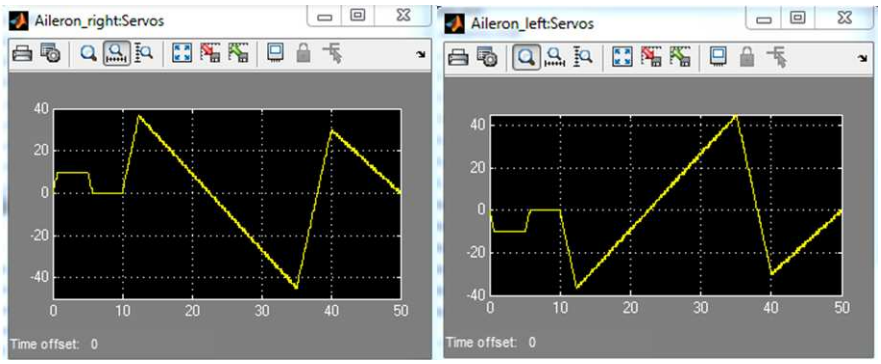

Figura 64: Resultado visual dos valores enviados aos ailerons.

5.7 Avaliação Comparativa do Processo de Transformação

Após a descrição dos exemplos em que é utilizado o processo de transformação para a abordagem SyMPLES, esta seção apresenta uma avaliação do mesmo. Esta análise foi realizada por meio da comparação das principais características do processo de transformação com outros trabalhos que propõem transformações entre modelos baseados em UML e Simulink. As características avaliadas são descritas a seguir.

Utilização de modelos de alto nível de abstração e da linguagem SysML

A abordagem SyMPLES utiliza a linguagem SysML para modelagem de sistemas embarcados. SysML foi a linguagem escolhida porque possibilita a representação de diferentes perspectivas da arquitetura de um sistema, por meio de requisitos, blocos, portas, diagramas paramétricos e alocações. Alguns trabalhos, como Perseil e Pautet (2010) e Zaki e Jawawi (2011) utilizam o perfil MARTE (*Modeling and Analysis of Real-time and Embedded systems*) para modelagem de sistemas embarcados. Contudo, MARTE se concentra nos aspectos de especificação de tempo-real, o que envolve diretivas para incluir nos modelos anotações necessárias para executar análises baseadas em tempo, como desempenho e escalabilidade.

Farkas et al. (2009) propõem uma abordagem em que é utilizada a linguagem UML para realizar a especificação de sistemas embarcados e a integração com blocos funcionais em

Simulink. No entanto, UML é uma linguagem de propósito geral e não considera conceitos importantes relacionados à engenharia de sistemas. A adoção da linguagem SysML na abordagem SyMPLES possibilitar explorar conceitos não contemplados na linguagem UML, como: blocos de parametrização; mapeamento entre requisitos e blocos; portas e fluxos.

Transformação de modelos baseados em UML para Simulink

Alguns trabalhos aplicam a transformação entre modelos UML e Simulink. No trabalho de (Sjöstedt et al. 2008) é realizada uma transformação de modelos Simulink para UML. Usando o modelo Simulink é gerado um artefato XML a partir da execução de um programa implementado na linguagem java. O arquivo XML é usado para gerar diagramas UML de atividades a partir de uma transformação ATL. Porém, o domínio aplicado é diferente do apresentado nessa dissertação e não é aplicado a uma PL como o SyMPLES.

No trabalho de Biehl, DeJiu, e Törngren (2010) é apresentado uma solução para o domínio de automóveis, em que é definido um processo de transformação bidirecional entre modelos Simulink e uma extensão UML chamada EAST-ADL. Também é desenvolvida uma ponte estrutural que mapeia os conceitos entre EAST-ADL e Simulink de forma que a semântica do modelo original é mantida. Porém o trabalho considera apenas o domínio de sistemas automotivos sem a aplicação de uma PL como definida pela abordagem SyMPLES.

No trabalho de (Brisolara 2007) é desenvolvido um processo de transformação UML para Simulink para diagramas de atividades, porém no trabalho dessa dissertação é usado diagramas de definição de estados, interno de blocos e máquina de estados e que fazem parte da abordagem SyMPLES.

5.8 Considerações Finais

Este capítulo apresentou uma avaliação sobre o processo de transformação que estende a abordagem SyMPLES e descreveu um exemplo de sua utilização. Com base no projeto Paparazzi a placa controladora do piloto automático Yapa 2 foi usada para representar os modelos usados para a transformação e simulação. O principal subsistema representado foi o controlador de rolagem que faz parte do sistema controlador de vôo. Exemplos com diagramas de definição de blocos, interno de blocos SysML e máquina de estados UML foram usados pelo processo de transformação.

Capítulo

6

Conclusão

Nesta dissertação foi apresentado um processo de transformação de modelos representados em SysML para modelos Simulink. O processo complementa a abordagem SyMPLES desenvolvida por Silva (2012) com ênfase na engenharia de aplicação de uma PL, e utiliza técnicas MDE de transformação de modelos baseada em metamodelos. Isso permite a geração de modelos específicos de plataforma de uma PL por meio do refinamento de abstrações de alto nível. Modelos SysML configurados de uma PL são usados como entrada para a transformação e representam sistemas embarcados nos níveis iniciais de desenvolvimento.

A abordagem de blocos funcionais é amplamente utilizada na indústria de sistemas embarcados. Para possibilitar o mapeamento entre os blocos SysML e blocos funcionais Simulink foi definido na abordagem SyMPLES um perfil para SysML denominado SyMPLES-ProfileFB. Por meio dos estereótipos definidos em tal perfil, é possível fornecer uma semântica adicional a um bloco padrão SysML. O processo de transformação foi avaliado usando uma arquitetura inicial da placa controladora Yapa 2 usada em VANTs Paparazzi.

6.1 Contribuições

As contribuições desta dissertação podem ser resumidas da seguinte forma:

- Um processo de transformação de modelos representados em SysML para modelos Simulink, utilizando o perfil SyMPLES-ProfileFB da abordagem SyMPLES;

- Uma extensão do perfil SyMPLES-ProfileFB a fim de permitir o mapeamento de elementos da biblioteca padrão e customizáveis;
- A implementação do mapeamento e transformação de diagramas SysML de definição de blocos, diagrama interno de blocos para blocos funcionais Simulink e o diagrama de estados para o bloco Stateflow.

Iniciar a especificação de um sistema em modelos de alto nível como SysML possibilita cobrir atividades complexas do desenvolvimento de sistemas embarcados. A abordagem MDE facilita o refinamento e transformação desses modelos. O processo de transformação de modelos é aplicado a arquitetura de uma PL para sistemas embarcados nas fases iniciais usando modelos mais abstratos SysML, que permite um melhor gerenciamento das variabilidades e representação baseada nos requisitos. Os modelos transformados permitem representar o sistema com blocos funcionais, que facilita a geração de código.

6.2 Limitações

As limitações do processo de transformação são:

- A transformação considera apenas diagramas SysML de definição de bloco, diagramas internos de blocos e diagrama de estados. Diagramas de atividades e diagramas paramétricos devem ser avaliados;
- O SyMPLES-ProfileFB inclui apenas alguns blocos funcionais basicos da biblioteca Simulink, relacionados ao desenvolvimento de VANTs. Porém, novos estereótipos podem ser adicionados ao SyMPLES-ProfileFB que permite gerar novas regras de mapeamento;
- O mapeamento de elementos dos diagramas SysML para Simulink é restrito. Apenas alguns elementos dos diagramas SysML são utilizados. Por exemplo, portas do tipo *Inout* não foram são mapeadas. Assim é necessário avaliar mais elementos para melhorar o mapeamento;
- A abordagem foi avaliada apenas para o domínio de VANTs. Uma avaliação mais completa é necessária.

6.3 Trabalhos futuros

Os trabalhos futuros estão relacionados com as limitações descritas na seção anterior. A seguir são apresentadas as principais direções de trabalhos futuros:

- Utilizar mais recursos da linguagem SysML para o mapeamento de elementos dos diagramas, assim como a adição de novos diagramas;
- Estender o SyMPLES-ProfileFB a fim de mapear mais elementos e representar mais atributos que podem ser transformados em parâmetros dos blocos funcionais Simulink;
- Desenvolver a especificação do modelo de blocos funcionais refinado para geração de código. Essa etapa pode ser concretizada com um modelo Simulink completo e específico de um VANT.

Referências

Almeida, P. 2008. p.108 “MDA – Model Driven Architecture: Improving Software Development Productivity in Large-Scale Enterprise Applications.” University of Fribourg, Switzerland. http://diuf.unifr.ch/main/softeng/teaching/studentprojects/dealmeida.

Arduino. 2012. “Arduino platform.” http://www.arduino.cc/.

Atkinson, C., and T. Kuhne. 2003. “Model-driven development: a metamodeling foundation.” *IEEE Software* 20(5): p. 36–41. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1231149 (October 4, 2012).

Bassi, L., C. Secchi, M. Bonfe, and C. Fantuzzi. 2011. “A SysML-Based Methodology for Manufacturing Machinery Modeling and Design.” *IEEE/ASME Transactions on Mechatronics* 16(6): p. 1049–1062. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5604318 (November 8, 2012).

Beuche, D. 2003. *Variant Management with pure::variants*. http://www.pure-systems.com/pure_variants.49.0.html.

Biehl, M., D. Chen, and M. Törngren. 2010. “Integrating safety analysis into the model-based development toolchain of automotive embedded systems.” *ACM SIGPLAN Notices* 45(4): p.125. http://portal.acm.org/citation.cfm?doid=1755951.1755907.

Biehl, M., C. Sjöstedt, and M. Törngren. 2010. “A modular tool integration approach : experiences from two case studies.” *3rd Workshop on Model-driven tool and Process Integration (MDTPI2010).*

Braga, R., K. Branco, O. Junior, and I. Gimenes. 2011. “Evolving Tiriba Design towards a Product line of Small Eletric-Powered UAVs.” In *1st Brazilian Conference on Critical Embedded Systems*, p. 63–72.

Brisolara, L. B. 2007. p.129 “Strategies for Embedded Software Development Based on High-level Models.” UFRGS - Porto Alegre.

Cantor, M.. 2003. “Rational Unified Process for Systems Engineering.” *The Rational Edge - IBM*: pp.17.

Coleman, C., J. Funk, J. Salvati, C. Whipple, T. Padir, and A. Wyglinski. 2012. p.180 *Design of an Autonomous Platform for Search and Rescue UAV Networks*.

Czarnecki, K., M. Antkiewicz, C. Hwan, P. Kim, S. Lau, and K. Pietroszek. 2005. “Model-driven software product lines.” In *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications - OOPSLA ’05*, New York, New York, USA: ACM Press, p. 126. http://portal.acm.org/citation.cfm?doid=1094855.1094896 (December 28, 2012).

Czarnecki, K., and S. Helsen. 2003. “Classification of Model Transformation Approaches.” In *OOPSLA’03 Workshop on the Generative Techniques in the Context Of Model-Driven Architecture*, Anaheim, California, USA, p. 17.

Douglass, B. P. 1997. p. 400 *Real-Time UML: Developing Efficient Objects for Embedded Systems*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Douglass, B. P. 2008. “The Telelogic Harmony/ESW Process for Real-Time and Embedded Development.” *Telelogic*: p.8.

Drones, D.. 2012. “Arduplane - Ardupilot-mega.” http://code.google.com/p/ardupilot-mega/.

Ebert, C., and C. Jones. 2009. “Embedded Software: Facts, Figures, and Future.” *Computer* 42(4): p.42–52. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5054871 (December 28, 2012).

EL-Khoury, J. 2006. p. 54 “A Model Management and Integration Platform for Mechatronics Product Development.” Royal Institute of Technology, KTH.

EMF. 2012. “Eclipse Modeling Framework Project.” http://www.eclipse.org/modeling/emf/.

Enac. 2003. “Paparazzi Project.” http://Paparazzi.enac.fr/wiki/Main_Page.

Estefan, J. A. 2008. “Survey of Model-Based Systems Engineering (MBSE) Methodologies.” *International Council on Systems Engineering (INCOSE)*: p.70.

FAA, Federal Aviation Administration. 2008. “Flight Controls.” In *Pilot’s Handbook of Aeronautical Knowledge*, p. 12.

Farkas, T., E. Meiseki, C. Neumann, K. Okano, A. Hinnerichs, and S. Kamiya. 2009. “Integration of UML with Simulink into embedded software engineering.” In *ICCAS-SICE*, p. 474 – 479.

Ferreira, R., L. Brisolara, J. C. B. Mattos, E. Spech, and E. Cota. 2009. p. 26 *Behavioral Modeling for Embedded Systems and Technologies*. eds. Luís Gomes and João M. Fernandes. IGI Global. http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-750-8 (December 28, 2012).

Fragal, V., E. A. Oliveira Junior, and I. M. S. Gimenes. 2011. “Mapping Software Product Line Features to Unmanned Aerial Vehicle Models.” In *1st Brazilian Conference on Critical Embedded Systems*, p. 49–54.

Friedenthal, S., R. Steiner, and A. C. Moore. 2008. p. 576 *Practical Guide to SysML: The Systems Modeling Language (The Mk/OMG Press)*. illustrate. Elsevier Science. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0123743796.

Heverhagen, T. 2003. “Integration of languages for programmable controllers into the Unified Modeling Language through Function Block Adapters.” University of Duisburg-Essen.

Hunter, A. 2012. “Graphical Modeling Framework (GMF) Notation.” http://www.eclipse.org/projects/project.php?id=modeling.gmp.gmf-notation.

INCOSE. 2006. “Object Oriented System Engineering Method.” *OOSEM Descriptive Outline for INCOSE SE Handbook Version 3*: p.6.

INCT-SEC. 2012. “Instituto Nacional de Ciência e Tecnologia.” http://www.inct-sec.org/br/.

Kleppe, A. G., J. B. Warmer, and W. Bast. 2003. 192 *MDA Explained: The Model Driven Architecture : Practice and Promise*. Addison-Wesley. http://books.google.com.br/books?id=-uodwVPBG6YC.

Linden, F., K. Schmif, and E. Rommes. 2007. 353 *Software Product Lines in Action*. Springer.

Louhichi, S., M. Graiet, M. Kmimech, M. T. Bhiri, W. Gaaloul, and E. Cariou. 2011. “ATL Transformation for the Generation of SCA Model.” *2011 Seventh International Conference on Semantics, Knowledge and Grids*: p.164–167. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6088108 (May 6, 2012).

Lykins, F. M. 2000. “Adapting UML for an Object-Oriented Systems Engineering Method (OOSEM).” In *INCOSE International Symposium*,.

Marwedel, P. 2010. p. 421 *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Springer; 2nd ed. 2011 edition (December 3, 2010).

MathWorks. 2012. “Stateflow.” http://www.mathworks.com/products/stateflow/.

Mellor, S. J. 2004. p. 150 *Mda Distilled: Principles of Model-Driven Architecture*. Addison-Wesley. http://books.google.com.br/books?id=LGzS1uiUa7AC.

Mellor, S. J., and M. J. Balcer. 2002. p. 416 *Executable Uml: A Foundation for Model-Driven Architecture*. Addison-Wesley. http://books.google.com.br/books?id=zBS0aWNjBqcC.

Mellor, S. J., A. N. Clark, and T. Futagami. 2003. “Model-driven development - Guest editor’s introduction.” *IEEE Software* 20(5): p.14–18. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1231145 (October 10, 2012).

Miller, J., and J. Mukerji. 2003. “MDA Guide Version 1.0.1.” (June): p.62. http://www.omg.org/cgi-bin/doc?omg/03-06-01.

Obeo. 2006. “Atlas Transformation Language.” http://www.obeo.fr/pages/atl-pro/en.

Oliveira, E. A. Júnior., I. M. S. Gimenes, and J. C. Maldonado. 2010. “Systematic Management of Variability in UML-based Software Product Lines.” *Journal of Universal Computer Science* 16: p. 2374–2393.

OMG. 2006. “Diagram Interchange.” *OMG*: p.86. http://www.omg.org/cgi-bin/doc?formal/06-04-04 (October 11, 2012).

Papyrus. 2012. “Open Source Tool for Graphical UML2 Modelling.” http://www.papyrusuml.org/scripts/home/publigen/content/templates/show.asp?P=128&L=EN&ITEMID=12.

Pastor, E., J. Lopez, and P. Royo. 2006. “An Embedded Architecture for Mission Control of Unmanned Aerial Vehicles.” In *9th EUROMICRO Conference on Digital System Design (DSD’06)*, IEEE, p. 554–560. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1690087 (December 12, 2012).

Perseil, I., and L. Pautet. 2010. “High-Level Abstraction Modeling for Detailed Analysis of Avionic Real-Time Systems.” In *2010 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems*, IEEE, p. 418–424. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5457742 (December 28, 2012).

Polzer, A., S. Kowalewski, and G. Botterweck. 2009. “Applying software product line techniques in model-based embedded systems engineering.” In *2009 ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, IEEE, p. 2–10. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5069132 (December 3, 2012).

Pratt, R. 2000. p. 382 *Flight control systems*. American Institute of Aeronautics and Astronautics. http://books.google.com.br/books?id=AYRTAAAAMAAJ.

Rothenbertg, J. 1989. A Rand note; N-3027-DARPA p.18 *The Nature of Modeling*. Santa Monica, California. www.rand.org/pubs/notes/2007/N3027.pdf.

Ruscio, D. 2007. p.118 “Specification of model transformation and weaving in model driven engineering.” Universit`a di L’Aquila. http://www.di.univaq.it/diruscio/PhDThesis_DiRuscio.pdf.

Schmidt, D.C. 2006. “Guest Editor’s Introduction: Model-Driven Engineering.” *Computer* 39(2): p.25–31. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1597083 (October 7, 2012).

Selic, B. 2003. “The pragmatics of model-driven development.” *IEEE Software* 20(5): p.19–25. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1231146 (October 9, 2012).

Shi, J.. 2007. “Model and Tool Integration in High Level Design of Embedded Systems.”

Silva, R. F. 2012. p.108 “SyMPLES : Uma Abordagem de Desenvolvimento de Linha de Produto para Sistemas Embarcados baseada em SysML.” Dissertação de mestrado, UEM.

Simulink. 1994. “Simulation and Model-Based Design.” http://www.mathworks.com/products/simulink/.

Simulink Coder. 2012. “Real-Time Workshop.” http://www.mathworks.com/products/simulink-coder/index.html.

Sjöstedt, C., M. Törngren, J. Shi, D. Chen, and V. Ahlsten. 2008. “Mapping Simulink to UML in the design of embedded systems:Investigating scenarios and transformations.” In *OMER4 Post-proceedings, 2008*, p. 137–160.

Steiner, E. M.. 2012. p. 102 “Gerenciamento de configuração de uma linha de produtos de software de veículos aéreos não tripulados.” Dissertação de mestrado, USP.

SysML. 2008. “OMG Systems Modeling Language.” *OMG*: p.234.

TOPCASED. 2012. “The Open-Source Toolkit for Critical Systems.” http://www.topcased.org/.

UML. 2005. “Unified Modeling Language.” *Introduction to OMG UML*. http://www.uml.org/.

YAPA. 2011. “YetAnotherPaparazziAutopilot v2.” http://Paparazzi.enac.fr/wiki/YAPA/v2.0.

Zaki, M. Z. M., and D. N. A. Jawawi. 2011. “Model-based methodology for implementing MARTE in embedded real-time software.” In *2011 IEEE Symposium on Computers & Informatics*, IEEE, p. 536–541. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5958973 (December 28, 2012).

Apêndice A

Este apêndice contém o script matlab que usa as APIs Simulink e stateflow do estudo de caso apresentado para o Yapa 2.

```
function new_uav15(modelname)
 if nargin == 0
    modelname = 'new_uav15';
    RQhWK8B30EeKNSu6wH1bjLA = 'new_uav15/Payload';
    RWJuIsB30EeKNSu6wH1bjLA = 'new_uav15/UAV';
    RXQb5IB30EeKNSu6wH1bjLA = 'new_uav15/Navigation';
    Rb22qoB30EeKNSu6wH1bjLA = 'new_uav15/Flight Control';
    RhpglwB30EeKNSu6wH1bjLA = 'new_uav15/Actuators';
    RikEwkB30EeKNSu6wH1bjLA = 'new_uav15/Servos';
    RkuL3sB30EeKNSu6wH1bjLA = 'new_uav15/Sensors';
    RodpzEB30EeKNSu6wH1bjLA = 'new_uav15/Camera';
    RTRf0B4JEeKNSu6wH1bjLA = 'new_uav15/Barometer';
    RlyMDl4EeKcJ4TEKT5BRw = 'new_uav15/Communication';
    RnOjsDsJEeKxfmhQd8wzw = 'new_uav15/Navigation/Yapa2:Autopilot';
    Rikt2QCEUEeKMCvIEM15I8A = 'new_uav15/Flight Control/loop:Roll Controller';
    RCcEIYCnTEeKD4vESWQFXyg = 'new_uav15/Flight Control/cmdDemux:Demux';
    RiAAQQCnTEeKD4vESWQFXyg = 'new_uav15/Flight Control/ServoCtrl:Mux';
    RS7MOID7YEeKtx69ukyHZJQ = 'new_uav15/Flight Control/loop:Pitch Controller';
    RXGoT0D7YEeKtx69ukyHZJQ = 'new_uav15/Flight Control/loop:Throttle Controller';
    Rr65VACBwEeKMPAvFvsPcA = 'new_uav15/Actuators/Aileron_left:Servos';
    RwybrsCBwEeKMPAvFvsPcA = 'new_uav15/Actuators/elevator:Servos';
    RDTm4gCgfEeKoDSkLDcRiQ = 'new_uav15/Actuators/rudder:Servos';
    Ro26YUCnAEeKD4vESWQFXyg = 'new_uav15/Actuators/Aileron_right:Servos';
    RfbGfACndEeKD4vESWQFXyg = 'new_uav15/Actuators/demuxServ:Demux';
    RDR1eMD7ZEeKtx69ukyHZJQ = 'new_uav15/Actuators/motor:Servos';
    RtpOsCBdEeKMPAvFvsPcA = 'new_uav15/Sensors/BPM085:Barometer';
    RONQXECgpEeKoDSkLDcRiQ = 'new_uav15/Sensors/part1:Batery';
    Rve3sDsHEeKxfmhQd8wzw = 'new_uav15/Sensors/MT3329:GPS';
    RzCzMDsHEeKxfmhQd8wzw = 'new_uav15/Sensors/MP6000:IMU';
    RfzxZYCQZEeKPk4amLu4W5w = 'new_uav15/Payload/enableCamera:CameraActivator';
    RtthXcCQZEeKPk4amLu4W5w = 'new_uav15/Payload/video1:VideoDisplay';
    RxgDbgCqIEeKGv79DEq1CwA = 'new_uav15/Payload/RGB:Camera';
    RFFp2QD1aEeKJ1dShKCcz2w = 'new_uav15/Communication/Desired Waypoint:Constant';
    RJ3z9sD1aEeKJ1dShKCcz2w = 'new_uav15/Communication/Desired Altitude:Constant';
    RPz794D1aEeKJ1dShKCcz2w = 'new_uav15/Communication/Desired Pitch Angle:Constant';
    RUKIiED1aEeKJ1dShKCcz2w = 'new_uav15/Communication/Desired Airspeed:Constant';
    RiDYw4D1aEeKJ1dShKCcz2w = 'new_uav15/Communication/m1:Mux';
    RZnrasD4cEeKs1LXrmbRGKw = 'new_uav15/Communication/receiver:XBee';

end
open_system(new_system(modelname));
add_block('built-in/SubSystem', RQhWK8B30EeKNSu6wH1bjLA);
set_param(RQhWK8B30EeKNSu6wH1bjLA, 'Position', [800 385 905 437]);
RQzBcICmtEeKD4vESWQFXyg = strcat(RQhWK8B30EeKNSu6wH1bjLA, '/payload_activator_0');
add_block('built-in/Inport', RQzBcICmtEeKD4vESWQFXyg);
RPL9xgDmBEeKcJ4TEKT5BRw = strcat(RQhWK8B30EeKNSu6wH1bjLA, '/Camera_OUT_1');
add_block('built-in/Outport', RPL9xgDmBEeKcJ4TEKT5BRw);

add_block('built-in/SubSystem', RXQb5IB30EeKNSu6wH1bjLA);
```

```
set_param(RXQb5IB30EeKNSu6wH1bjLA, 'Position', [655 290 761 376]);
RI44kCgiEeKoDSkLDcRiQ = strcat(RXQb5IB30EeKNSu6wH1bjLA, '/GPS_0');
add_block('built-in/Inport', RI44kCgiEeKoDSkLDcRiQ);
RgKr8YCHbEeKsL8IoZhbiyw = strcat(RXQb5IB30EeKNSu6wH1bjLA, '/Navigation Cmds_1');
add_block('built-in/Outport', RgKr8YCHbEeKsL8IoZhbiyw);
RPwAmkCkWEeK74PyobivnqQ = strcat(RXQb5IB30EeKNSu6wH1bjLA, '/Pressure_2');
add_block('built-in/Inport', RPwAmkCkWEeK74PyobivnqQ);
RTuM6cCkWEeK74PyobivnqQ = strcat(RXQb5IB30EeKNSu6wH1bjLA, '/Batery_3');
add_block('built-in/Inport', RTuM6cCkWEeK74PyobivnqQ);
RRhqn4CkKEeK74PyobivnqQ = strcat(RXQb5IB30EeKNSu6wH1bjLA, '/PayloadActivator_4');
add_block('built-in/Outport', RRhqn4CkKEeK74PyobivnqQ);
Ro9F4IDlEeKcJ4TEKT5BRw = strcat(RXQb5IB30EeKNSu6wH1bjLA, '/IMU_5');
add_block('built-in/Inport', Ro9F4IDlEeKcJ4TEKT5BRw);
RpGCjUDmAEeKcJ4TEKT5BRw = strcat(RXQb5IB30EeKNSu6wH1bjLA, '/Mission data_6');
add_block('built-in/Inport', RpGCjUDmAEeKcJ4TEKT5BRw);

add_block('built-in/SubSystem', Rb22qoB30EeKNSu6wH1bjLA);
set_param(Rb22qoB30EeKNSu6wH1bjLA, 'Position', [860 295 966 346]);
ROXDToCgmEeKoDSkLDcRiQ = strcat(Rb22qoB30EeKNSu6wH1bjLA, '/NavCmm_0');
add_block('built-in/Inport', ROXDToCgmEeKoDSkLDcRiQ);
RAj5sCnBEeKD4vESWQFXyg = strcat(Rb22qoB30EeKNSu6wH1bjLA, '/Servo_Cmd_1');
add_block('built-in/Outport', RAj5sCnBEeKD4vESWQFXyg);

add_block('built-in/SubSystem', RhpglwB30EeKNSu6wH1bjLA);
set_param(RhpglwB30EeKNSu6wH1bjLA, 'Position', [1040 295 1141 346]);
RvffO0CnAEeKD4vESWQFXyg = strcat(RhpglwB30EeKNSu6wH1bjLA, '/ServoData_0');
add_block('built-in/Inport', RvffO0CnAEeKD4vESWQFXyg);

add_block('built-in/SubSystem', RkuL3sB30EeKNSu6wH1bjLA);
set_param(RkuL3sB30EeKNSu6wH1bjLA, 'Position', [425 300 531 376]);
RDaaJgCduEeKH4dsWRuHDXw = strcat(RkuL3sB30EeKNSu6wH1bjLA, '/GPS Data_0');
add_block('built-in/Outport', RDaaJgCduEeKH4dsWRuHDXw);
RdOfn8Cj1EeK74PyobivnqQ = strcat(RkuL3sB30EeKNSu6wH1bjLA, '/Pressure Data_1');
add_block('built-in/Outport', RdOfn8Cj1EeK74PyobivnqQ);
Rei4A8Cj1EeK74PyobivnqQ = strcat(RkuL3sB30EeKNSu6wH1bjLA, '/Batery charge_2');
add_block('built-in/Outport', Rei4A8Cj1EeK74PyobivnqQ);
RP5AP8Dl8EeKcJ4TEKT5BRw = strcat(RkuL3sB30EeKNSu6wH1bjLA, '/IMU_3');
add_block('built-in/Outport', RP5AP8Dl8EeKcJ4TEKT5BRw);

add_block('built-in/SubSystem', RlyMDl4EeKcJ4TEKT5BRw);
set_param(RlyMDl4EeKcJ4TEKT5BRw, 'Position', [430 200 541 251]);
Ru17TwDlEeKcJ4TEKT5BRw = strcat(RlyMDl4EeKcJ4TEKT5BRw, '/Nav_cmd_0');
add_block('built-in/Inport', Ru17TwDlEeKcJ4TEKT5BRw);
Rw0lY0DlEeKcJ4TEKT5BRw = strcat(RlyMDl4EeKcJ4TEKT5BRw, '/Received Data_1');
add_block('built-in/Outport', Rw0lY0DlEeKcJ4TEKT5BRw);
RaD8AoDmBEeKcJ4TEKT5BRw = strcat(RlyMDl4EeKcJ4TEKT5BRw, '/Payload_2');
add_block('built-in/Inport', RaD8AoDmBEeKcJ4TEKT5BRw);

add_line('new_uav15', 'Sensors/1', 'Navigation/1','autorouting','on');
add_line('new_uav15', 'Sensors/2', 'Navigation/2','autorouting','on');
add_line('new_uav15', 'Sensors/3', 'Navigation/3','autorouting','on');
add_line('new_uav15', 'Navigation/1', 'Flight Control/1','autorouting','on');
add_line('new_uav15', 'Navigation/2', 'Payload/1','autorouting','on');
add_line('new_uav15', 'Flight Control/1', 'Actuators/1','autorouting','on');
add_line('new_uav15', 'Sensors/4', 'Navigation/4','autorouting','on');
add_line('new_uav15', 'Communication/1', 'Navigation/5','autorouting','on');
add_line('new_uav15', 'Navigation/1', 'Communication/1','autorouting','on');
add_line('new_uav15', 'Payload/1', 'Communication/2','autorouting','on');
add_block('built-in/SubSystem', RnOjsDsJEeKxfmhQd8wzw);
set_param(RnOjsDsJEeKxfmhQd8wzw, 'Position', [235 9 393 138]);
RRyM7AD1SEeKJ1dShKCcz2w = strcat(RnOjsDsJEeKxfmhQd8wzw, '/flowport1_0');
add_block('built-in/Inport', RRyM7AD1SEeKJ1dShKCcz2w);
RTTkAED1SEeKJ1dShKCcz2w = strcat(RnOjsDsJEeKxfmhQd8wzw, '/flowport2_1');
add_block('built-in/Inport', RTTkAED1SEeKJ1dShKCcz2w);
RUCI9UD1SEeKJ1dShKCcz2w = strcat(RnOjsDsJEeKxfmhQd8wzw, '/flowport3_2');
add_block('built-in/Inport', RUCI9UD1SEeKJ1dShKCcz2w);
RVWb2wD1SEeKJ1dShKCcz2w = strcat(RnOjsDsJEeKxfmhQd8wzw, '/flowport4_3');
add_block('built-in/Inport', RVWb2wD1SEeKJ1dShKCcz2w);
RV7fsD1SEeKJ1dShKCcz2w = strcat(RnOjsDsJEeKxfmhQd8wzw, '/flowport5_4');
add_block('built-in/Outport', RV7fsD1SEeKJ1dShKCcz2w);
RWgX6UD1SEeKJ1dShKCcz2w = strcat(RnOjsDsJEeKxfmhQd8wzw, '/flowport6_5');
add_block('built-in/Outport', RWgX6UD1SEeKJ1dShKCcz2w);
RJ8E4D1ZEeKJ1dShKCcz2w = strcat(RnOjsDsJEeKxfmhQd8wzw, '/flowport7_6');
add_block('built-in/Inport', RJ8E4D1ZEeKJ1dShKCcz2w);
```

```
set_param(RXQb5IB30EeKNSu6wH1bjLA,'location',[90 55 746 426]);
set_param(RgKr8YCHbEeKsL8IoZhbiyw, 'Position', [686 90 706 110]);
set_param(RI44kCgiEeKoDSkLDcRiQ, 'Position', [10 45 30 65]);
set_param(RRhqn4CkKEeK74PyobivnqQ, 'Position', [686 140 706 160]);
set_param(RPwAmkCkWEeK74PyobivnqQ, 'Position', [10 70 30 90]);
set_param(RTuM6cCkWEeK74PyobivnqQ, 'Position', [10 95 30 115]);
set_param(Ro9F4IDlEeKcJ4TEKT5BRw, 'Position', [10 130 30 150]);
set_param(RpGCjUDmAEeKcJ4TEKT5BRw, 'Position', [10 160 30 180]);
add_line('new_uav15/Navigation', 'Yapa2:Autopilot/1',
'PayloadActivator_4/1','autorouting','on');
add_line('new_uav15/Navigation', 'Yapa2:Autopilot/2', 'Navigation
Cmds_1/1','autorouting','on');
add_line('new_uav15/Navigation', 'GPS_0/1', 'Yapa2:Autopilot/1','autorouting','on');
add_line('new_uav15/Navigation', 'Pressure_2/1', 'Yapa2:Autopilot/2','autorouting','on');
add_line('new_uav15/Navigation', 'Batery_3/1', 'Yapa2:Autopilot/3','autorouting','on');
add_line('new_uav15/Navigation', 'IMU_5/1', 'Yapa2:Autopilot/4','autorouting','on');
add_line('new_uav15/Navigation', 'Mission data_6/1', 'Yapa2:Autopilot/5','autorouting','on');
add_block('built-in/SubSystem', Rikt2QCEUEeKMCvIEM15I8A);
set_param(Rikt2QCEUEeKMCvIEM15I8A, 'Position', [270 24 441 165]);
RwgyvkCEUEeKMCvIEM15I8A = strcat(Rikt2QCEUEeKMCvIEM15I8A, '/Aileron_L_0');
add_block('built-in/Outport', RwgyvkCEUEeKMCvIEM15I8A);
RWQykCEUEeKMCvIEM15I8A = strcat(Rikt2QCEUEeKMCvIEM15I8A, '/Roll_1');
add_block('built-in/Inport', RWQykCEUEeKMCvIEM15I8A);
Rlr78CEUEeKMCvIEM15I8A = strcat(Rikt2QCEUEeKMCvIEM15I8A, '/Yaw_2');
add_block('built-in/Inport', Rlr78CEUEeKMCvIEM15I8A);
RjttT0CkFEeK74PyobivnqQ = strcat(Rikt2QCEUEeKMCvIEM15I8A, '/Rudder_3');
add_block('built-in/Outport', RjttT0CkFEeK74PyobivnqQ);
RzkvcCnBEeKD4vESWQFXyg = strcat(Rikt2QCEUEeKMCvIEM15I8A, '/Aileron_R_4');
add_block('built-in/Outport', RzkvcCnBEeKD4vESWQFXyg);


sfnew('state_machine');
rt = sfroot;
m = rt.find('-isa', 'Simulink.BlockDiagram', 'Name', 'state_machine');
ch = m.find('-isa','Stateflow.Chart');

sm_1 = Stateflow.Data(ch(1));
sm_1.set('Scope', 'Input');
sm_1.set('Name', 'Roll');
sm_1 = Stateflow.Data(ch(1));
sm_1.set('Scope', 'Output');
sm_1.set('Name', 'Aileron_R');
sm_1 = Stateflow.Data(ch(1));
sm_1.set('Scope', 'Output');
sm_1.set('Name', 'Aileron_L');
sm_1 = Stateflow.Data(ch(1));
sm_1.set('Scope', 'Input');
sm_1.set('Name', 'Yaw');
sm_1 = Stateflow.Data(ch(1));
sm_1.set('Scope', 'Output');
sm_1.set('Name', 'Rudder');
sm_RDwOEcCmEeKD4vESWQFXyg = Stateflow.State(ch(1));
sm_str = sprintf('roll_controller');
sm_RDwOEcCmEeKD4vESWQFXyg.LabelString = sm_str;
sm_RDwOEcCmEeKD4vESWQFXyg.Position = [41 45 990 211];
sm_RLeYqACmEeKD4vESWQFXyg = Stateflow.State(ch(1));
sm_str = sprintf('Init \n  Aileron_L=0;Aileron_T-0;');
sm_RLeYqACmEeKD4vESWQFXyg.LabelString = sm_str;
sm_RLeYqACmEeKD4vESWQFXyg.Position = [155 51 200 41];
sm_RQxe7UCmEeKD4vESWQFXyg = Stateflow.State(ch(1));
sm_str = sprintf('roll left \n Aileron_L=Aileron_L+1;Aileron_R=Aileron_R-1;');
sm_RQxe7UCmEeKD4vESWQFXyg.LabelString = sm_str;
sm_RQxe7UCmEeKD4vESWQFXyg.Position = [175 156 40 40];
sm_RVhuoICnHEeKD4vESWQFXyg = Stateflow.State(ch(1));
sm_str = sprintf('roll_right \n Aileron_L=Aileron_L-1;Aileron_R=Aileron_R+1;');
sm_RVhuoICnHEeKD4vESWQFXyg.LabelString = sm_str;
sm_RVhuoICnHEeKD4vESWQFXyg.Position = [570 116 40 40];
sm_Rx8jhoCnVEeKD4vESWQFXyg = Stateflow.State(ch(1));
sm_str = sprintf('ReadValue');
sm_Rx8jhoCnVEeKD4vESWQFXyg.LabelString = sm_str;
sm_Rx8jhoCnVEeKD4vESWQFXyg.Position = [430 51 72 50];
sm_RDwOEcCmEeKD4vESWQFXyg.IsGrouped = true;
sm_RDwOEcCmEeKD4vESWQFXyg.Chart.Decomposition = 'PARALLEL_AND';
sm_RxWXXUD79EeKtx69ukyHZJQ = Stateflow.State(ch(1));
sm_str = sprintf('yaw_controller');
sm_RxWXXUD79EeKtx69ukyHZJQ.LabelString = sm_str;
```

```
sm_RxWXXUD79EeKtx69ukyHZJQ.Position = [41 256 990 220];
sm_RTuvOUD8BEeKtx69ukyHZJQ = Stateflow.State(ch(1));
sm_str = sprintf('Init \n  Rudder=0;');
sm_RTuvOUD8BEeKtx69ukyHZJQ.LabelString = sm_str;
sm_RTuvOUD8BEeKtx69ukyHZJQ.Position = [135 292 101 42];
sm_RYmGl4D8BEeKtx69ukyHZJQ = Stateflow.State(ch(1));
sm_str = sprintf('ReadValue');
sm_RYmGl4D8BEeKtx69ukyHZJQ.LabelString = sm_str;
sm_RYmGl4D8BEeKtx69ukyHZJQ.Position = [378 295 77 38];
sm_RZBlA8D8BEeKtx69ukyHZJQ = Stateflow.State(ch(1));
sm_str = sprintf('yaw_left \n Rudder=Rudder-1;');
sm_RZBlA8D8BEeKtx69ukyHZJQ.LabelString = sm_str;
sm_RZBlA8D8BEeKtx69ukyHZJQ.Position = [297 386 40 40];
sm_RZZOr8D8BEeKtx69ukyHZJQ = Stateflow.State(ch(1));
sm_str = sprintf('yaw_right \n Rudder=Rudder+1;');
sm_RZZOr8D8BEeKtx69ukyHZJQ.LabelString = sm_str;
sm_RZZOr8D8BEeKtx69ukyHZJQ.Position = [573 361 40 40];
sm_RxWXXUD79EeKtx69ukyHZJQ.IsGrouped = true;

sm_RQJLrkCnGEeKD4vESWQFXyg = Stateflow.Transition(ch(1));
sm_RQJLrkCnGEeKD4vESWQFXyg.Destination = sm_RLeYqACmEeKD4vESWQFXyg;
sm_RQJLrkCnGEeKD4vESWQFXyg.DestinationOClock = 0;
xsource = sm_RLeYqACmEeKD4vESWQFXyg.Position(1);
ysource = sm_RLeYqACmEeKD4vESWQFXyg.Position(2) -20;
sm_RQJLrkCnGEeKD4vESWQFXyg.SourceEndPoint = [xsource ysource];

sm_R6YUCnVEeKD4vESWQFXyg = Stateflow.Transition(ch(1));
sm_R6YUCnVEeKD4vESWQFXyg.Source = sm_RLeYqACmEeKD4vESWQFXyg;
sm_R6YUCnVEeKD4vESWQFXyg.Destination = sm_Rx8jhoCnVEeKD4vESWQFXyg;
sm_R6YUCnVEeKD4vESWQFXyg.SourceOClock = 7;
sm_R6YUCnVEeKD4vESWQFXyg.DestinationOClock = 2;

sm_RyyrYCnVEeKD4vESWQFXyg = Stateflow.Transition(ch(1));
sm_RyyrYCnVEeKD4vESWQFXyg.LabelString = '[(Roll<45 && Aileron_R<Roll)]';
sm_RyyrYCnVEeKD4vESWQFXyg.Source = sm_Rx8jhoCnVEeKD4vESWQFXyg;
sm_RyyrYCnVEeKD4vESWQFXyg.Destination = sm_RVhuoICnHEeKD4vESWQFXyg;
sm_RyyrYCnVEeKD4vESWQFXyg.SourceOClock = 5;
sm_RyyrYCnVEeKD4vESWQFXyg.DestinationOClock = 10;

sm_RIFcZcCnYEeKD4vESWQFXyg = Stateflow.Transition(ch(1));
sm_RIFcZcCnYEeKD4vESWQFXyg.LabelString = '[(Roll>-45 && Aileron_R>Roll)]';
sm_RIFcZcCnYEeKD4vESWQFXyg.Source = sm_Rx8jhoCnVEeKD4vESWQFXyg;
sm_RIFcZcCnYEeKD4vESWQFXyg.Destination = sm_RQxe7UCmEeKD4vESWQFXyg;
sm_RIFcZcCnYEeKD4vESWQFXyg.SourceOClock = 7;
sm_RIFcZcCnYEeKD4vESWQFXyg.DestinationOClock = 2;

sm_RPwRMCnYEeKD4vESWQFXyg = Stateflow.Transition(ch(1));
sm_RPwRMCnYEeKD4vESWQFXyg.LabelString = '[after(0.01,sec)]';
sm_RPwRMCnYEeKD4vESWQFXyg.Source = sm_RVhuoICnHEeKD4vESWQFXyg;
sm_RPwRMCnYEeKD4vESWQFXyg.Destination = sm_Rx8jhoCnVEeKD4vESWQFXyg;
sm_RPwRMCnYEeKD4vESWQFXyg.SourceOClock = 7;
sm_RPwRMCnYEeKD4vESWQFXyg.DestinationOClock = 2;

sm_RSwpX4CnYEeKD4vESWQFXyg = Stateflow.Transition(ch(1));
sm_RSwpX4CnYEeKD4vESWQFXyg.LabelString = '[after(0.01,sec)]';
sm_RSwpX4CnYEeKD4vESWQFXyg.Source = sm_RQxe7UCmEeKD4vESWQFXyg;
sm_RSwpX4CnYEeKD4vESWQFXyg.Destination = sm_Rx8jhoCnVEeKD4vESWQFXyg;
sm_RSwpX4CnYEeKD4vESWQFXyg.SourceOClock = 10;
sm_RSwpX4CnYEeKD4vESWQFXyg.DestinationOClock = 5;

sm_RpUkHQD8BEeKtx69ukyHZJQ = Stateflow.Transition(ch(1));
sm_RpUkHQD8BEeKtx69ukyHZJQ.Destination = sm_RTuvOUD8BEeKtx69ukyHZJQ;
sm_RpUkHQD8BEeKtx69ukyHZJQ.DestinationOClock = 0;
xsource = sm_RTuvOUD8BEeKtx69ukyHZJQ.Position(1);
ysource = sm_RTuvOUD8BEeKtx69ukyHZJQ.Position(2) -20;
sm_RpUkHQD8BEeKtx69ukyHZJQ.SourceEndPoint = [xsource ysource];

sm_Rp0wDMD8BEeKtx69ukyHZJQ = Stateflow.Transition(ch(1));
sm_Rp0wDMD8BEeKtx69ukyHZJQ.Source = sm_RTuvOUD8BEeKtx69ukyHZJQ;
sm_Rp0wDMD8BEeKtx69ukyHZJQ.Destination = sm_RYmGl4D8BEeKtx69ukyHZJQ;
sm_Rp0wDMD8BEeKtx69ukyHZJQ.SourceOClock = 5;
sm_Rp0wDMD8BEeKtx69ukyHZJQ.DestinationOClock = 10;

sm_RrCfBoD8BEeKtx69ukyHZJQ = Stateflow.Transition(ch(1));
sm_RrCfBoD8BEeKtx69ukyHZJQ.LabelString = '[(Yaw<45 && Rudder<Yaw)]';
```

```
sm_RrCfBoD8BEeKtx69ukyHZJQ.Source = sm_RYmGl4D8BEeKtx69ukyHZJQ;
sm_RrCfBoD8BEeKtx69ukyHZJQ.Destination = sm_RZZOr8D8BEeKtx69ukyHZJQ;
sm_RrCfBoD8BEeKtx69ukyHZJQ.SourceOClock = 5;
sm_RrCfBoD8BEeKtx69ukyHZJQ.DestinationOClock = 10;

sm_RrZsAsD8BEeKtx69ukyHZJQ = Stateflow.Transition(ch(1));
sm_RrZsAsD8BEeKtx69ukyHZJQ.LabelString = '[after(0.01,sec)]';
sm_RrZsAsD8BEeKtx69ukyHZJQ.Source = sm_RZZOr8D8BEeKtx69ukyHZJQ;
sm_RrZsAsD8BEeKtx69ukyHZJQ.Destination = sm_RYmGl4D8BEeKtx69ukyHZJQ;
sm_RrZsAsD8BEeKtx69ukyHZJQ.SourceOClock = 10;
sm_RrZsAsD8BEeKtx69ukyHZJQ.DestinationOClock = 5;

sm_RrwNDQD8BEeKtx69ukyHZJQ = Stateflow.Transition(ch(1));
sm_RrwNDQD8BEeKtx69ukyHZJQ.LabelString = '[(Yaw>-45 && Rudder>Yaw)]';
sm_RrwNDQD8BEeKtx69ukyHZJQ.Source = sm_RYmGl4D8BEeKtx69ukyHZJQ;
sm_RrwNDQD8BEeKtx69ukyHZJQ.Destination = sm_RZBlA8D8BEeKtx69ukyHZJQ;
sm_RrwNDQD8BEeKtx69ukyHZJQ.SourceOClock = 7;
sm_RrwNDQD8BEeKtx69ukyHZJQ.DestinationOClock = 2;

sm_RsEfRMD8BEeKtx69ukyHZJQ = Stateflow.Transition(ch(1));
sm_RsEfRMD8BEeKtx69ukyHZJQ.LabelString = '[after(0.01,sec)]';
sm_RsEfRMD8BEeKtx69ukyHZJQ.Source = sm_RZBlA8D8BEeKtx69ukyHZJQ;
sm_RsEfRMD8BEeKtx69ukyHZJQ.Destination = sm_RYmGl4D8BEeKtx69ukyHZJQ;
sm_RsEfRMD8BEeKtx69ukyHZJQ.SourceOClock = 2;
sm_RsEfRMD8BEeKtx69ukyHZJQ.DestinationOClock = 7;

add_block('state_machine/Chart', strcat(Rikt2QCEUEeKMCvIEM15I8A, '/Roll control'));

bdclose('state_machine');
add_block('built-in/Demux', RCcEIYCnTEeKD4vESWQFXyg);
set_param(RCcEIYCnTEeKD4vESWQFXyg, 'Position', [80 89 148 129]);
set_param(RCcEIYCnTEeKD4vESWQFXyg, 'Outputs', '4');

add_block('built-in/Mux', RiAAQQCnTEeKD4vESWQFXyg);
set_param(RiAAQQCnTEeKD4vESWQFXyg, 'Position', [590 89 640 137]);
set_param(RiAAQQCnTEeKD4vESWQFXyg, 'Inputs', '5');

add_block('built-in/SubSystem', RS7MOID7YEeKtx69ukyHZJQ);
set_param(RS7MOID7YEeKtx69ukyHZJQ, 'Position', [265 214 441 265]);
RP1iPUD7ZEeKtx69ukyHZJQ = strcat(RS7MOID7YEeKtx69ukyHZJQ, '/Pitch_0');
add_block('built-in/Inport', RP1iPUD7ZEeKtx69ukyHZJQ);
RQTmEUD7ZEeKtx69ukyHZJQ = strcat(RS7MOID7YEeKtx69ukyHZJQ, '/flowport2_1');
add_block('built-in/Outport', RQTmEUD7ZEeKtx69ukyHZJQ);

add_block('built-in/SubSystem', RXGoT0D7YEeKtx69ukyHZJQ);
set_param(RXGoT0D7YEeKtx69ukyHZJQ, 'Position', [265 274 431 325]);
ROoKdQD7ZEeKtx69ukyHZJQ = strcat(RXGoT0D7YEeKtx69ukyHZJQ, '/throttle_0');
add_block('built-in/Inport', ROoKdQD7ZEeKtx69ukyHZJQ);
RPR4vsD7ZEeKtx69ukyHZJQ = strcat(RXGoT0D7YEeKtx69ukyHZJQ, '/flowport2_1');
add_block('built-in/Outport', RPR4vsD7ZEeKtx69ukyHZJQ);

set_param(Rb22qoB30EeKNSu6wH1bjLA,'location',[35 25 941 496]);
set_param(ROXDToCgmEeKoDSkLDcRiQ, 'Position', [10 150 30 170]);
set_param(RAj5sCnBEeKD4vESWQFXyg, 'Position', [881 155 901 175]);
add_line('new_uav15/Flight Control', 'NavCmm_0/1', 'cmdDemux:Demux/1','autorouting','on');
add_line('new_uav15/Flight Control', 'loop:Roll Controller/3',
'ServoCtrl:Mux/1','autorouting','on');
add_line('new_uav15/Flight Control', 'loop:Roll Controller/1',
'ServoCtrl:Mux/2','autorouting','on');
add_line('new_uav15/Flight Control', 'loop:Roll Controller/2',
'ServoCtrl:Mux/3','autorouting','on');
add_line('new_uav15/Flight Control', 'ServoCtrl:Mux/1', 'Servo_Cmd_1/1','autorouting','on');
add_line('new_uav15/Flight Control', 'cmdDemux:Demux/1', 'loop:Roll
Controller/1','autorouting','on');
add_line('new_uav15/Flight Control', 'cmdDemux:Demux/2', 'loop:Roll
Controller/2','autorouting','on');
add_line('new_uav15/Flight Control', 'cmdDemux:Demux/3', 'loop:Pitch
Controller/1','autorouting','on');
add_line('new_uav15/Flight Control', 'cmdDemux:Demux/4', 'loop:Throttle
Controller/1','autorouting','on');
add_line('new_uav15/Flight Control', 'loop:Pitch Controller/1',
'ServoCtrl:Mux/4','autorouting','on');
add_line('new_uav15/Flight Control', 'loop:Throttle Controller/1',
'ServoCtrl:Mux/5','autorouting','on');
add_block('built-in/Scope', Rr65VACBwEeKMPAvFvsPcA);
```

```
set_param(Rr65VACBwEeKMPAvFvsPcA, 'Position', [420 94 488 119]);

add_block('built-in/Scope', RwybrsCBwEeKMPAvFvsPcA);
set_param(RwybrsCBwEeKMPAvFvsPcA, 'Position', [420 224 495 249]);

add_block('built-in/Scope', RDTm4gCgfEeKoDSkLDcRiQ);
set_param(RDTm4gCgfEeKoDSkLDcRiQ, 'Position', [420 159 493 184]);

add_block('built-in/Scope', Ro26YUCnAEeKD4vESWQFXyg);
set_param(Ro26YUCnAEeKD4vESWQFXyg, 'Position', [420 24 485 49]);

add_block('built-in/Demux', RfbGfACndEeKD4vESWQFXyg);
set_param(RfbGfACndEeKD4vESWQFXyg, 'Position', [130 104 210 152]);
set_param(RfbGfACndEeKD4vESWQFXyg, 'Outputs', '5');

add_block('built-in/Scope', RDR1eMD7ZEeKtx69ukyHZJQ);
set_param(RDR1eMD7ZEeKtx69ukyHZJQ, 'Position', [420 279 491 304]);

set_param(RhpglwB30EeKNSu6wH1bjLA,'location',[35 30 891 451]);
set_param(RvffO0CnAEeKD4vESWQFXyg, 'Position', [10 165 30 185]);
add_line('new_uav15/Actuators', 'ServoData_0/1', 'demuxServ:Demux/1','autorouting','on');
add_line('new_uav15/Actuators', 'demuxServ:Demux/1',
'Aileron_right:Servos/1','autorouting','on');
add_line('new_uav15/Actuators', 'demuxServ:Demux/2',
'Aileron_left:Servos/1','autorouting','on');
add_line('new_uav15/Actuators', 'demuxServ:Demux/3', 'rudder:Servos/1','autorouting','on');
add_line('new_uav15/Actuators', 'demuxServ:Demux/4', 'elevator:Servos/1','autorouting','on');
add_line('new_uav15/Actuators', 'demuxServ:Demux/5', 'motor:Servos/1','autorouting','on');
add_block('built-in/SubSystem', RtpOsCBdEeKMPAvFvsPcA);
set_param(RtpOsCBdEeKMPAvFvsPcA, 'Position', [235 139 411 190]);
RdHn4CBuEeKMPAvFvsPcA = strcat(RtpOsCBdEeKMPAvFvsPcA, '/flowport1_0');
add_block('built-in/Outport', RdHn4CBuEeKMPAvFvsPcA);

add_block('built-in/Constant', RONQXECgpEeKoDSkLDcRiQ);
set_param(RONQXECgpEeKoDSkLDcRiQ, 'Value', '10');
set_param(RONQXECgpEeKoDSkLDcRiQ, 'Position', [275 264 345 289]);

add_block('built-in/Reference', Rve3sDsHEeKxfmhQd8wzw);
set_param(Rve3sDsHEeKxfmhQd8wzw, 'SourceBlock', 'GPS/MT3329:GPS');
set_param(Rve3sDsHEeKxfmhQd8wzw, 'Position', [285 19 347 45]);

add_block('built-in/Reference', RzCzMDsHEeKxfmhQd8wzw);
set_param(RzCzMDsHEeKxfmhQd8wzw, 'SourceBlock', 'aerolibnav/Three-axis Inertial Measurement
Unit');
set_param(RzCzMDsHEeKxfmhQd8wzw, 'Position', [290 194 351 220]);

set_param(RkuL3sB30EeKNSu6wH1bjLA,'location',[75 40 771 436]);
set_param(RDaaJgCduEeKH4dsWRuHDXw, 'Position', [711 75 731 95]);
set_param(RdOfn8Cj1EeK74PyobivnqQ, 'Position', [711 146 731 166]);
set_param(Rei4A8Cj1EeK74PyobivnqQ, 'Position', [711 320 731 340]);
set_param(RP5AP8Dl8EeKcJ4TEKT5BRw, 'Position', [711 255 731 275]);
add_line('new_uav15/Sensors', 'part1:Batery/1', 'Batery charge_2/1','autorouting','on');
add_line('new_uav15/Sensors', 'BPM085:Barometer/1', 'Pressure Data_1/1','autorouting','on');
add_line('new_uav15/Sensors', 'MT3329:GPS/1', 'GPS Data_0/1','autorouting','on');
add_line('new_uav15/Sensors', 'MP6000:IMU/1', 'IMU_3/1','autorouting','on');
add_block('built-in/SubSystem', RfzxZYCQZEeKPk4amLu4W5w);
RfzxZYCQZEeKPk4amLu4W5w_p = strcat(RfzxZYCQZEeKPk4amLu4W5w, '/Enable');
add_block('built-in/EnablePort', RfzxZYCQZEeKPk4amLu4W5w_p, 'Position', [225, 20, 245, 40]);
set_param(RfzxZYCQZEeKPk4amLu4W5w,'location',[40 40 550 300]);
set_param(RfzxZYCQZEeKPk4amLu4W5w, 'Position', [420 159 513 184]);
RaMkq4CQaEeKPk4amLu4W5w = strcat(RfzxZYCQZEeKPk4amLu4W5w, '/flowport1_0');
add_block('built-in/Inport', RaMkq4CQaEeKPk4amLu4W5w);
RdWTYCQaEeKPk4amLu4W5w = strcat(RfzxZYCQZEeKPk4amLu4W5w, '/flowport2_1');
add_block('built-in/Outport', RdWTYCQaEeKPk4amLu4W5w);

set_param(RdWTYCQaEeKPk4amLu4W5w, 'Position', [490 101 510 121]);
set_param(RaMkq4CQaEeKPk4amLu4W5w, 'Position', [10 106 30 126]);
add_line('new_uav15/Payload/enableCamera:CameraActivator', 'flowport1_0/1',
'flowport2_1/1','autorouting','on');
add_block('built-in/Reference', RtthXcCQZEeKPk4amLu4W5w);
set_param(RtthXcCQZEeKPk4amLu4W5w, 'SourceBlock', 'visionsinks/To Video Display');
set_param(RtthXcCQZEeKPk4amLu4W5w, 'Position', [660 159 738 183]);

add_block('built-in/Reference', RxgDbgCqIEeKGv79DEq1CwA);
```

```
set_param(RxgDbgCqIEeKGv79DEq1CwA, 'SourceBlock', 'imaqlib/From Video Device');
set_param(RxgDbgCqIEeKGv79DEq1CwA, 'Position', [175 124 263 149]);

set_param(RQhWK8B30EeKNSu6wH1bjLA,'location',[40 60 966 416]);
set_param(RQzBcICmtEeKD4vESWQFXyg, 'Position', [10 82 30 102]);
set_param(RPL9xgDmBEeKcJ4TEKT5BRw, 'Position', [906 113 926 133]);
add_line('new_uav15/Payload', 'enableCamera:CameraActivator/1',
'video1:VideoDisplay/1','autorouting','on');
add_line('new_uav15/Payload', 'payload_activator_0/1',
'enableCamera:CameraActivator/Enable','autorouting','on');
add_line('new_uav15/Payload', 'RGB:Camera/1',
'enableCamera:CameraActivator/1','autorouting','on');
add_line('new_uav15/Payload', 'enableCamera:CameraActivator/1',
'Camera_OUT_1/1','autorouting','on');
add_block('built-in/Constant', RFFp2QD1aEeKJ1dShKCcz2w);
set_param(RFFp2QD1aEeKJ1dShKCcz2w, 'Value', '2');
set_param(RFFp2QD1aEeKJ1dShKCcz2w, 'Position', [510 39 602 63]);

add_block('built-in/Constant', RJ3z9sD1aEeKJ1dShKCcz2w);
set_param(RJ3z9sD1aEeKJ1dShKCcz2w, 'Value', '3');
set_param(RJ3z9sD1aEeKJ1dShKCcz2w, 'Position', [515 94 606 119]);

add_block('built-in/Constant', RPz794D1aEeKJ1dShKCcz2w);
set_param(RPz794D1aEeKJ1dShKCcz2w, 'Value', '5');
set_param(RPz794D1aEeKJ1dShKCcz2w, 'Position', [510 204 603 229]);

add_block('built-in/Constant', RUKIiED1aEeKJ1dShKCcz2w);
set_param(RUKIiED1aEeKJ1dShKCcz2w, 'Value', '4');
set_param(RUKIiED1aEeKJ1dShKCcz2w, 'Position', [525 149 610 174]);

add_block('built-in/Mux', RiDYw4D1aEeKJ1dShKCcz2w);
set_param(RiDYw4D1aEeKJ1dShKCcz2w, 'Position', [790 124 841 163]);
set_param(RiDYw4D1aEeKJ1dShKCcz2w, 'Inputs', '4');

add_block('built-in/SubSystem', RZnrasD4cEeKs1LXrmbRGKw);
set_param(RZnrasD4cEeKs1LXrmbRGKw, 'Position', [175 94 311 145]);
RlR5TUD8SEeK4ZeEFKYWaA = strcat(RZnrasD4cEeKs1LXrmbRGKw, '/flowport1_0');
add_block('built-in/Outport', RlR5TUD8SEeK4ZeEFKYWaA);

set_param(RlyMDl4EeKcJ4TEKT5BRw,'location',[60 35 1151 496]);
set_param(Ru17TwDlEeKcJ4TEKT5BRw, 'Position', [10 335 30 355]);
set_param(Rw0lY0DlEeKcJ4TEKT5BRw, 'Position', [1091 193 1111 213]);
set_param(RaD8AoDmBEeKcJ4TEKT5BRw, 'Position', [10 370 30 390]);
add_line('new_uav15/Communication', 'Desired Waypoint:Constant/1',
'm1:Mux/1','autorouting','on');
add_line('new_uav15/Communication', 'Desired Altitude:Constant/1',
'm1:Mux/2','autorouting','on');
add_line('new_uav15/Communication', 'Desired Airspeed:Constant/1',
'm1:Mux/3','autorouting','on');
add_line('new_uav15/Communication', 'Desired Pitch Angle:Constant/1',
'm1:Mux/4','autorouting','on');
add_line('new_uav15/Communication', 'm1:Mux/1', 'Received Data_1/1','autorouting','on');
set_param(modelname,'location',[300 300 1500 1000]);

save_system(modelname);
```

Apêndice B

Este apendice contém o código XMI que representa o metamodelo modificado Simulink usado em transformações ATL.

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="Simulink"
    nsURI="http:///se.kth.md.attest2/Simulink/3.0" nsPrefix="sim">
  <eClassifiers xsi:type="ecore:EClass" name="ProtoObject" abstract="true"
eSuperTypes="platform:/plugin/org.eclipse.emf.ecore/model/Ecore.ecore#//EObject">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="simulinkName" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="position" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="uuid" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Port" eSuperTypes="#//ProtoObject">
    <eStructuralFeatures xsi:type="ecore:EReference" name="connections" upperBound="-1"
        eType="#//Line" eOpposite="#//Line/source"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Inport" eSuperTypes="#//Port">
    <eStructuralFeatures xsi:type="ecore:EReference" name="parent" lowerBound="1"
        eType="#//System" eOpposite="#//System/inports"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Outport" eSuperTypes="#//Port">
    <eStructuralFeatures xsi:type="ecore:EReference" name="parent" lowerBound="1"
        eType="#//System" eOpposite="#//System/outports"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Line" eSuperTypes="#//ProtoObject">
    <eStructuralFeatures xsi:type="ecore:EReference" name="source" unique="false"
        lowerBound="1" eType="#//Port" eOpposite="#//Port/connections"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="destination" unique="false"
        lowerBound="1" eType="#//Port"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="simuNameSrc" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="simuNameDst" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="source_uml" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="destination_uml" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="System" eSuperTypes="#//ProtoObject">
    <eStructuralFeatures xsi:type="ecore:EReference" name="children" upperBound="-1"
        eType="#//SystemReference" containment="true" eOpposite="#//SystemReference/parent"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="lines" upperBound="-1"
```

```
        eType="#//Line" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="inports" upperBound="-1"
        eType="#//Inport" containment="true" eOpposite="#//Inport/parent"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="outports" upperBound="-1"
        eType="#//Outport" containment="true" eOpposite="#//Outport/parent"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="ports" upperBound="-1"
        eType="#//Port" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="filename" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Model" eSuperTypes="#//ProtoObject">
    <eStructuralFeatures xsi:type="ecore:EReference" name="parts" upperBound="-1"
        eType="#//System" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="root" eType="#//System"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="lines" upperBound="-1"
        eType="#//Line" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SystemReference" eSuperTypes="#//ProtoObject #//System">
    <eStructuralFeatures xsi:type="ecore:EReference" name="target" lowerBound="1"
        eType="#//System"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="parent" lowerBound="1"
        eType="#//System" eOpposite="#//System/children"/>
  </eClassifiers>
</ecore:EPackage>
```

Apêndice C

Este apendice mostra as regras geradas para a transformação ATL.

```
-- @atlcompiler atl2006
-- @nsURI UML=http://www.eclipse.org/uml2/2.1.0/UML


module SysML2Simulink;
create OUT: Simulink from IN: NOTATION, IN2: UML;

rule Diagram2Model {
	from
		s1: NOTATION!Diagram --(s1 = thisModule.ValidSYSMLDiagram)

	to
		t1: Simulink!Model (
			name <- s1.name,
			simulinkName <- s1.element.getPath(),
			type <- s1.type,
			parts <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Shape)) ->
					select(e | not e.element.oclIsTypeOf(UML!Comment)) -> collect(e |
					thisModule.Shape2System(e)),
			lines <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Connector)) ->
					select(e | e.element.oclIsTypeOf(UML!Usage)) -> collect(e |
					thisModule.Edge2Line(e))
		)
}

lazy rule Shape2System {
	from
		s1: NOTATION!Shape
	to
		t1: Simulink!System (
			name <- s1.element.name,
			simulinkName <- s1.element.eContainer().getID(),
			type <- s1.element.eClass().name,
			uuid <- s1.element.getID(),
			position <- s1.layoutConstraint.getBounds(),
			children <- if (s1.eContents() -> select(e | e.
					oclIsTypeOf(NOTATION!BasicCompartment)) -> size() > 0) then
					(thisModule.BasicCompartment2SystemReference(s1.eContents() ->
							select(e | e.oclIsTypeOf(NOTATION!BasicCompartment)) ->
							first())
				)
				else
					(OclUndefined)
				endif,
			ports <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Shape)) ->
					select(e | e.element.oclIsTypeOf(UML!Port)) -> collect(e | thisModule.
					Shape2Port(e, 'model_type')),
			lines <- s1.eContainer().eContents() -> select(e | e.
					oclIsTypeOf(NOTATION!Connector)) -> select(e | e.element.
					oclIsTypeOf(UML!Connector)) -> collect(e | thisModule.Edge2Line(e)),
			-- statemachine rule calls
			children <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Shape)) ->
					select(e | e.element.oclIsTypeOf(UML!Pseudostate)) -> collect(e |
					thisModule.Shape2SystemReference_state(e)),
			children <- s1.eContents() -> select(e | e.
					oclIsTypeOf(NOTATION!DecorationNode)) -> select(e | e.eContainer().
					element.oclIsTypeOf(UML!StateMachine)) -> collect(e | e.eContents()
					-> select(i | i.oclIsTypeOf(NOTATION!Shape)) -> select(i | i.element.
					oclIsTypeOf(UML!Region)) -> collect(i | thisModule.
					Shape2SystemReference_state_recursive(i))),
			lines <- s1.eContainer().eContents() -> select(e | e.
					oclIsTypeOf(NOTATION!Connector)) -> select(e | e.element.
					oclIsTypeOf(UML!Transition)) -> collect(e | thisModule.
					Edge2Transition(e))
		)
}
```

```
lazy rule Shape2Port {
	from
		s1: NOTATION!Shape,
		s2: String
	to
		t1: Simulink!Port (
			name <- s1.element.name,
			uuid <- s1.element.getID(),
			position <- s1.layoutConstraint.getBounds(),
			type <- s2
		)
}

lazy rule BasicCompartment2SystemReference {
	from
		s1: NOTATION!BasicCompartment
	to
		t1: Simulink!SystemReference (
			name <- s1.type,
			children <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Shape)) ->
					select(e | e.element.oclIsTypeOf(UML!Property)) -> collect(e |
					thisModule.Shape2SystemReference(e))
		)
}

lazy rule Shape2SystemReference {
	from
		s1: NOTATION!Shape
	to
		t1: Simulink!SystemReference (
			name <- s1.element.name + ':' + s1.element.type.name,
			type <- s1.element.type.getID(),
			uuid <- s1.element.getID(),
			position <- s1.layoutConstraint.getBounds(),
			ports <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Shape)) ->
					select(e | e.element.oclIsTypeOf(UML!Port)) -> collect(e | thisModule.
					Shape2Port(e, 'internal_type'))
		)
}

lazy rule Edge2Line {
	from
		i: NOTATION!Edge
	to
		o: Simulink!Line (
			name <- i.element.name,
			type <- i.element.eClass().name,
			--simuNameSrc <- i.element.end->first().getConnectorLink(),
			--simuNameDst <- i.element.end->last().getConnectorLink(),
			source_uml <- if (i.element.oclIsTypeOf(UML!Connector)) then
					(i.element.end -> first().role.getID())
				else
					(i.element.client -> first().getID())
				endif,
			destination_uml <- if (i.element.oclIsTypeOf(UML!Connector)) then
					(i.element.end -> last().role.getID())
				else
					(i.element.supplier -> last().getID())
				endif
		)
}

-- state machine rules
lazy rule Edge2Transition {
	from
		i: NOTATION!Edge
	to
		o: Simulink!Line (
			name <- i.element.name,
			type <- i.element.eClass().name,
			uuid <- i.element.getID(),
			source_uml <- i.element.source.getID(),
			destination_uml <- i.element.target.getID(),
			position <- i.bendpoints.points -> collect(e | e.toString()) -> first()
		)
}

lazy rule Shape2SystemReference_state {
	from
		s1: NOTATION!Shape
	to
		t1: Simulink!SystemReference (
			name <- s1.element.name,
			type <- if (s1.element.oclIsTypeOf(UML!Pseudostate)) then
					(if (not s1.element.kind.oclIsUndefined()) then
						(s1.element.eClass().name + ':' + s1.element.kind)
					else
						(s1.element.eClass().name)
					endif)
				else
					(s1.element.eClass().name)
				endif,
			uuid <- s1.element.getID(),
			position <- s1.layoutConstraint.getBounds(),
			children <- s1.eContents() -> select(e | e.
					oclIsTypeOf(NOTATION!DecorationNode)) -> select(e | e.eContainer().
					element.oclIsTypeOf(UML!State)) -> collect(e | e.eContents() ->
					select(i | i.oclIsTypeOf(NOTATION!Shape)) -> select(i | i.element.
```

```
                                                  oclIsTypeOf(UML!Region)) -> collect(i | thisModule.
                                                  Shape2SystemReference_state_recursive(i)))
                    )
}

lazy rule Shape2SystemReference_state_recursive {
          from
                    s1: NOTATION!Shape
          to
                    t1: Simulink!SystemReference (
                              name <- s1.element.name,
                              type <- s1.element.eClass().name,
                              uuid <- s1.element.getID(),
                              position <- s1.layoutConstraint.getBounds(),
                              children <- s1.eContents() -> select(e | e.
                                                  oclIsTypeOf(NOTATION!DecorationNode)) -> select(e | e.eContainer().
                                                  element.oclIsTypeOf(UML!Region)) -> collect(e | thisModule.
                                                  Shape2SystemReference_region_states(e))
                    )
}

lazy rule Shape2SystemReference_region_states {
          from
                    s1: NOTATION!Shape
          to
                    t1: Simulink!SystemReference (
                              children <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Shape)) ->
                                                  select(e | e.element.oclIsTypeOf(UML!State)) -> collect(e |
                                                  thisModule.Shape2SystemReference_state(e)),
                              children <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Shape)) ->
                                                  select(e | e.element.oclIsTypeOf(UML!Pseudostate)) -> collect(e |
                                                  thisModule.Shape2SystemReference_state(e)),
                              children <- s1.eContents() -> select(e | e.oclIsTypeOf(NOTATION!Shape)) ->
                                                  select(e | e.element.oclIsTypeOf(UML!FinalState)) -> collect(e |
                                                  thisModule.Shape2SystemReference_state(e))
                    )
}

--helpers
helper context OclAny def: getPath(): String =
          if (not self.oclIsUndefined()) then
                    (self.name + '/' + self.eContainer().getPath())
          else
                    ('')
          endif;

helper context NOTATION!Bounds def: getBounds(): String =
          'x=' + if(not self.x.oclIsUndefined())then(self.x)else('0')endif + ' y=' + if(not
                              self.y.oclIsUndefined())then(self.y)else('0')endif + ' width=' + if(not self.
                              width.oclIsUndefined())then(self.width)else('20')endif + ' height=' + if(not
                              self.height.oclIsUndefined())then(self.height)else('20')endif;

helper context OclAny def: getID(): String =
          if (not self.oclIsUndefined()) then
                    (self.eResource().getURIFragment(self))
          else
                    (OclUndefined)
          endif;

helper context UML!ConnectorEnd def: getConnectorLink(): String =
          if (not self.partWithPort.oclIsUndefined()) then
                    (self.partWithPort.name)
          else
                    ('')
          endif;
```

Apêndice D

Este apendice mostra o codigo XMI gerado pela transformação ATL para o exemplo do Yapa 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:sim="http:///se.kth.md.attest2/Simulink/3.0">
  <sim:Model name="NewDiagram" simulinkName="SysMLmodel/" type="BlockDefinition">
    <parts name="Payload" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=800 y=385 width=105 height=52"
uuid="_QhWK8B30EeKNSu6wH1bjLA" type="Class">
      <ports name="payload_activator" position="x=-10 y=13 width=20 height=20" uuid="_QzBcICmtEeKD4vESWQFXyg" type="model_type"/>
      <ports name="Camera_OUT" position="x=95 y=16 width=20 height=20" uuid="_PL9xgDmBEeKcJ4TEKT5BRw" type="model_type"/>
    </parts>
    <parts name="UAV" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=755 y=185 width=106 height=36" uuid="_WJuIsB30EeKNSu6wH1bjLA"
type="Class"/>
    <parts name="Navigation" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=655 y=290 width=106 height=86"
uuid="_XQb5IB30EeKNSu6wH1bjLA" type="Class">
      <ports name="GPS" position="x=-10 y=5 width=20 height=20" uuid="_6I44kCgiEeKoD_SkLDcRiQ" type="model_type"/>
      <ports name="Navigation Cmds" position="x=96 y=20 width=20 height=20" uuid="_gKr8YCHbEeKsL8IoZhbiyw" type="model_type"/>
      <ports name="Pressure" position="x=-10 y=25 width=20 height=20" uuid="_PwAmkCkWEeK74PyobivnqQ" type="model_type"/>
      <ports name="Batery" position="x=-10 y=45 width=20 height=20" uuid="_TuM6cCkWEeK74PyobivnqQ" type="model_type"/>
      <ports name="PayloadActivator" position="x=96 y=40 width=20 height=20" uuid="_Rhqn4CkKEeK74PyobivnqQ" type="model_type"/>
      <ports name="IMU" position="x=-10 y=70 width=20 height=20" uuid="_o9F4IDl_EeKcJ4TEKT5BRw" type="model_type"/>
      <ports name="Mission data" position="x=10 y=-10 width=20 height=20" uuid="_pGCjUDmAEeKcJ4TEKT5BRw" type="model_type"/>
    </parts>
    <parts name="Flight Control" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=860 y=295 width=106 height=51"
uuid="_b22qoB30EeKNSu6wH1bjLA" type="Class">
      <ports name="NavCmm" position="x=-10 y=25 width=20 height=20" uuid="_OXDToCgmEeKoD_SkLDcRiQ" type="model_type"/>
      <ports name="Servo_Cmd" position="x=96 y=15 width=20 height=20" uuid="_2Aj5sCnBEeKD4vESWQFXyg" type="model_type"/>
    </parts>
    <parts name="Actuators" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=1040 y=295 width=101 height=51"
uuid="_hpglwB30EeKNSu6wH1bjLA" type="Class">
      <ports name="ServoData" position="x=-10 y=15 width=20 height=20" uuid="_vffO0CnAEeKD4vESWQFXyg" type="model_type"/>
    </parts>
    <parts name="Servos" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=955 y=375 width=106 height=52" uuid="_ikEwkB30EeKNSu6wH1bjLA"
type="Class"/>
    <parts name="Sensors" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=425 y=300 width=106 height=76"
uuid="_kuL3sB30EeKNSu6wH1bjLA" type="Class">
      <ports name="GPS Data" position="x=96 y=-5 width=20 height=20" uuid="_DaaJgCduEeKH4dsWRuHDXw" type="model_type"/>
      <ports name="Pressure Data" position="x=96 y=15 width=20 height=20" uuid="_dOfn8Cj1EeK74PyobivnqQ" type="model_type"/>
      <ports name="Batery charge" position="x=96 y=35 width=20 height=20" uuid="_ei4A8Cj1EeK74PyobivnqQ" type="model_type"/>
      <ports name="IMU" position="x=96 y=60 width=20 height=20" uuid="_P5AP8Dl8EeKcJ4TEKT5BRw" type="model_type"/>
    </parts>
    <parts name="Camera" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=955 y=435 width=101 height=51" uuid="_odpzEB30EeKNSu6wH1bjLA"
type="Class"/>
    <parts name="Barometer" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=425 y=415 width=101 height=51"
uuid="_5TRf0B4JEeKNSu6wH1bjLA" type="Class"/>
    <parts name="Communication" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=430 y=200 width=111 height=51" uuid="_l-
y_MDl4EeKcJ4TEKT5BRw" type="Class">
      <ports name="Nav_cmd" position="x=101 y=0 width=20 height=20" uuid="_u17TwDl_EeKcJ4TEKT5BRw" type="model_type"/>
      <ports name="Received Data" position="x=101 y=25 width=20 height=20" uuid="_w0lY0Dl_EeKcJ4TEKT5BRw" type="model_type"/>
      <ports name="Payload" position="x=-10 y=19 width=20 height=20" uuid="_aD8AoDmBEeKcJ4TEKT5BRw" type="model_type"/>
    </parts>
    <lines name="Usage1" type="Usage" source_uml="_DaaJgCduEeKH4dsWRuHDXw" destination_uml="_6I44kCgiEeKoD_SkLDcRiQ"/>
    <lines name="Usage1" type="Usage" source_uml="_dOfn8Cj1EeK74PyobivnqQ" destination_uml="_PwAmkCkWEeK74PyobivnqQ"/>
    <lines name="Usage2" type="Usage" source_uml="_ei4A8Cj1EeK74PyobivnqQ" destination_uml="_TuM6cCkWEeK74PyobivnqQ"/>
    <lines name="Usage4" type="Usage" source_uml="_gKr8YCHbEeKsL8IoZhbiyw" destination_uml="_OXDToCgmEeKoD_SkLDcRiQ"/>
    <lines name="Usage5" type="Usage" source_uml="_Rhqn4CkKEeK74PyobivnqQ" destination_uml="_QzBcICmtEeKD4vESWQFXyg"/>
    <lines name="Usage9" type="Usage" source_uml="_2Aj5sCnBEeKD4vESWQFXyg" destination_uml="_vffO0CnAEeKD4vESWQFXyg"/>
    <lines name="Usage1" type="Usage" source_uml="_P5AP8Dl8EeKcJ4TEKT5BRw" destination_uml="_o9F4IDl_EeKcJ4TEKT5BRw"/>
    <lines name="Usage2" type="Usage" source_uml="_w0lY0Dl_EeKcJ4TEKT5BRw" destination_uml="_pGCjUDmAEeKcJ4TEKT5BRw"/>
    <lines name="Usage3" type="Usage" source_uml="_gKr8YCHbEeKsL8IoZhbiyw" destination_uml="_u17TwDl_EeKcJ4TEKT5BRw"/>
    <lines name="Usage4" type="Usage" source_uml="_PL9xgDmBEeKcJ4TEKT5BRw" destination_uml="_aD8AoDmBEeKcJ4TEKT5BRw"/>
  </sim:Model>
  <sim:Model name="D_Navigation_internal" simulinkName="Navigation/SysMLmodel/" type="InternalBlock">
    <parts name="Navigation" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=90 y=55 width=696 height=376"
uuid="_XQb5IB30EeKNSu6wH1bjLA" type="Class">
      <children name="compartment_sysml_structure">
        <children name="Yapa2:Autopilot" position="x=235 y=9 width=158 height=129" uuid="_4nOjsDsJEeKxf-mhQd8wzw" type="_7FVk4DsJEeKxf-
mhQd8wzw">
          <ports name="flowport1" position="x=-10 y=4 width=20 height=20" uuid="_RyM7AD1SEeKJ1dShKCcz2w" type="internal_type"/>
```

```
          <ports name="flowport2" position="x=-10 y=25 width=20 height=20" uuid="_TTkAED1SEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport3" position="x=-10 y=45 width=20 height=20" uuid="_UCI9UD1SEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport4" position="x=-10 y=72 width=20 height=20" uuid="_VWb2wD1SEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport5" position="x=148 y=86 width=20 height=20" uuid="_V_7fsD1SEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport6" position="x=148 y=42 width=20 height=20" uuid="_WgX6UD1SEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport7" position="x=-10 y=104 width=20 height=20" uuid="_2J8E4D1ZEeKJ1dShKCcz2w" type="internal_type"/>
        </children>
      </children>
      <lines name="connector1" type="Connector" source_uml="_V_7fsD1SEeKJ1dShKCcz2w" destination_uml="_Rhqn4CkKEeK74PyobivnqQ"/>
      <lines name="connector2" type="Connector" source_uml="_WgX6UD1SEeKJ1dShKCcz2w" destination_uml="_gKr8YCHbEeKsL8IoZhbiyw"/>
      <lines name="connector3" type="Connector" source_uml="_6I44kCgiEeKoD_SkLDcRiQ" destination_uml="_RyM7AD1SEeKJ1dShKCcz2w"/>
      <lines name="connector4" type="Connector" source_uml="_PwAmkCkWEeK74PyobivnqQ" destination_uml="_TTkAED1SEeKJ1dShKCcz2w"/>
      <lines name="connector5" type="Connector" source_uml="_TuM6cCkWEeK74PyobivnqQ" destination_uml="_UCI9UD1SEeKJ1dShKCcz2w"/>
      <lines name="connector6" type="Connector" source_uml="_o9F4IDl_EeKcJ4TEKT5BRw" destination_uml="_VWb2wD1SEeKJ1dShKCcz2w"/>
      <lines name="connector7" type="Connector" source_uml="_pGCjUDmAEeKcJ4TEKT5BRw" destination_uml="_2J8E4D1ZEeKJ1dShKCcz2w"/>
      <ports name="Navigation Cmds" position="x=686 y=90 width=20 height=20" uuid="_gKr8YCHbEeKsL8IoZhbiyw" type="model_type"/>
      <ports name="GPS" position="x=-10 y=45 width=20 height=20" uuid="_6I44kCgiEeKoD_SkLDcRiQ" type="model_type"/>
      <ports name="PayloadActivator" position="x=686 y=140 width=20 height=20" uuid="_Rhqn4CkKEeK74PyobivnqQ" type="model_type"/>
      <ports name="Pressure" position="x=-10 y=70 width=20 height=20" uuid="_PwAmkCkWEeK74PyobivnqQ" type="model_type"/>
      <ports name="Batery" position="x=-10 y=95 width=20 height=20" uuid="_TuM6cCkWEeK74PyobivnqQ" type="model_type"/>
      <ports name="IMU" position="x=-10 y=130 width=20 height=20" uuid="_o9F4IDl_EeKcJ4TEKT5BRw" type="model_type"/>
      <ports name="Mission data" position="x=-10 y=160 width=20 height=20" uuid="_pGCjUDmAEeKcJ4TEKT5BRw" type="model_type"/>
    </parts>
  </sim:Model>
  <sim:Model name="D_Flight_internal" simulinkName="Flight Control/SysMLmodel/" type="InternalBlock">
    <parts name="Flight Control" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=35 y=25 width=891 height=446"
uuid="_b22qoB30EeKNSu6wH1bjLA" type="Class">
      <children name="compartment_sysml_structure">
        <children name="loop:Roll Controller" position="x=270 y=24 width=171 height=141" uuid="_ikt2QCEUEeKMCvIEM15I8A"
type="_t4ScQCEUEeKMCvIEM15I8A">
          <ports name="Aileron_L" position="x=161 y=80 width=20 height=20" uuid="_wgyvkCEUEeKMCvIEM15I8A" type="internal_type"/>
          <ports name="Roll" position="x=-10 y=55 width=20 height=20" uuid="_3WQykCEUEeKMCvIEM15I8A" type="internal_type"/>
          <ports name="Yaw" position="x=-10 y=80 width=20 height=20" uuid="_6lr78CEUEeKMCvIEM15I8A" type="internal_type"/>
          <ports name="Rudder" position="x=161 y=105 width=20 height=20" uuid="_jttT0CkFEeK74PyobivnqQ" type="internal_type"/>
          <ports name="Aileron_R" position="x=161 y=60 width=20 height=20" uuid="_3zkvcCnBEeKD4vESWQFXyg" type="internal_type"/>
        </children>
        <children name="cmdDemux:Demux" position="x=80 y=89 width=136 height=81" uuid="_CcEIYCnTEeKD4vESWQFXyg"
type="_DR_DwCnTEeKD4vESWQFXyg">
          <ports name="flowport1" position="x=-10 y=20 width=20 height=20" uuid="_JveroCnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport2" position="x=126 y=-5 width=20 height=20" uuid="_K3T2YCnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport3" position="x=126 y=15 width=20 height=20" uuid="_MsXTgCnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport4" position="x=126 y=37 width=20 height=20" uuid="_PF-2QCnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport12" position="x=126 y=65 width=20 height=20" uuid="_TedTcD79EeKtx69ukyHZJQ" type="internal_type"/>
        </children>
        <children name="ServoCtrl:Mux" position="x=590 y=89 width=101 height=96" uuid="_iAAQQCnTEeKD4vESWQFXyg"
type="_ixqWsCnTEeKD4vESWQFXyg">
          <ports name="flowport1" position="x=-10 y=-5 width=20 height=20" uuid="_suyHICnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport2" position="x=-10 y=15 width=20 height=20" uuid="_tiXgMCnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport3" position="x=-10 y=35 width=20 height=20" uuid="_vfNnYCnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport4" position="x=-10 y=58 width=20 height=20" uuid="_yQ9WQCnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport5" position="x=91 y=25 width=20 height=20" uuid="_3F9WUCnTEeKD4vESWQFXyg" type="internal_type"/>
          <ports name="flowport21" position="x=-10 y=83 width=20 height=20" uuid="_e3PwoD79EeKtx69ukyHZJQ" type="internal_type"/>
        </children>
        <children name="loop:Pitch Controller" position="x=265 y=214 width=176 height=51" uuid="_S7MOID7YEeKtx69ukyHZJQ"
type="_UUOdoD7YEeKtx69ukyHZJQ">
          <ports name="Pitch" position="x=-10 y=22 width=20 height=20" uuid="_P1iPUD7ZEeKtx69ukyHZJQ" type="internal_type"/>
          <ports name="flowport2" position="x=166 y=15 width=20 height=20" uuid="_QTmEUD7ZEeKtx69ukyHZJQ" type="internal_type"/>
        </children>
        <children name="loop:Throttle Controller" position="x=265 y=274 width=166 height=51" uuid="_XGoT0D7YEeKtx69ukyHZJQ"
type="_YFxcoD7YEeKtx69ukyHZJQ">
          <ports name="throttle" position="x=-10 y=16 width=20 height=20" uuid="_OoKdQD7ZEeKtx69ukyHZJQ" type="internal_type"/>
          <ports name="flowport2" position="x=156 y=18 width=20 height=20" uuid="_PR4vsD7ZEeKtx69ukyHZJQ" type="internal_type"/>
        </children>
      </children>
      <lines name="connector7" type="Connector" source_uml="_OXDToCgmEeKoD_SkLDcRiQ" destination_uml="_JveroCnTEeKD4vESWQFXyg"/>
      <lines name="connector11" type="Connector" source_uml="_3zkvcCnBEeKD4vESWQFXyg" destination_uml="_suyHICnTEeKD4vESWQFXyg"/>
      <lines name="connector12" type="Connector" source_uml="_wgyvkCEUEeKMCvIEM15I8A" destination_uml="_tiXgMCnTEeKD4vESWQFXyg"/>
      <lines name="connector13" type="Connector" source_uml="_jttT0CkFEeK74PyobivnqQ" destination_uml="_vfNnYCnTEeKD4vESWQFXyg"/>
      <lines name="connector15" type="Connector" source_uml="_3F9WUCnTEeKD4vESWQFXyg" destination_uml="_2Aj5sCnBEeKD4vESWQFXyg"/>
      <lines name="connector16" type="Connector" source_uml="_K3T2YCnTEeKD4vESWQFXyg" destination_uml="_3WQykCEUEeKMCvIEM15I8A"/>
      <lines name="connector17" type="Connector" source_uml="_MsXTgCnTEeKD4vESWQFXyg" destination_uml="_6lr78CEUEeKMCvIEM15I8A"/>
      <lines name="connector18" type="Connector" source_uml="_PF-2QCnTEeKD4vESWQFXyg" destination_uml="_P1iPUD7ZEeKtx69ukyHZJQ"/>
      <lines name="connector19" type="Connector" source_uml="_TedTcD79EeKtx69ukyHZJQ" destination_uml="_OoKdQD7ZEeKtx69ukyHZJQ"/>
      <lines name="connector20" type="Connector" source_uml="_QTmEUD7ZEeKtx69ukyHZJQ" destination_uml="_yQ9WQCnTEeKD4vESWQFXyg"/>
      <lines name="connector21" type="Connector" source_uml="_PR4vsD7ZEeKtx69ukyHZJQ" destination_uml="_e3PwoD79EeKtx69ukyHZJQ"/>
      <ports name="NavCmm" position="x=-10 y=150 width=20 height=20" uuid="_OXDToCgmEeKoD_SkLDcRiQ" type="model_type"/>
      <ports name="Servo_Cmd" position="x=881 y=155 width=20 height=20" uuid="_2Aj5sCnBEeKD4vESWQFXyg" type="model_type"/>
    </parts>
  </sim:Model>
  <sim:Model name="D_Actuators_internal" simulinkName="Actuators/SysMLmodel/" type="InternalBlock">
    <parts name="Actuators" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=35 y=30 width=841 height=401"
uuid="_hpglwB30EeKNSu6wH1bjLA" type="Class">
      <children name="compartment_sysml_structure">
        <children name="Aileron_left:Servos" position="x=420 y=94 width=136 height=51" uuid="_r65VACBwEeK_MPAvFvsPcA"
type="_ikEwkB30EeKNSu6wH1bjLA">
          <ports name="flowport1" position="x=-10 y=12 width=20 height=20" uuid="_dA2FoCByEeK_MPAvFvsPcA" type="internal_type"/>
        </children>
        <children name="elevator:Servos" position="x=420 y=224 width=151 height=51" uuid="_wybrsCBwEeK_MPAvFvsPcA"
type="_ikEwkB30EeKNSu6wH1bjLA">
          <ports name="flowport2" position="x=-10 y=10 width=20 height=20" uuid="_dx5vICByEeK_MPAvFvsPcA" type="internal_type"/>
        </children>
        <children name="rudder:Servos" position="x=420 y=159 width=146 height=51" uuid="_DTm4gCgfEeKoD_SkLDcRiQ"
type="_ikEwkB30EeKNSu6wH1bjLA">
          <ports name="flowport5" position="x=-10 y=20 width=20 height=20" uuid="_J-8IECgfEeKoD_SkLDcRiQ" type="internal_type"/>
        </children>
        <children name="Aileron_right:Servos" position="x=420 y=24 width=131 height=51" uuid="_o26YUCnAEeKD4vESWQFXyg"
type="_ikEwkB30EeKNSu6wH1bjLA">
          <ports name="flowport6" position="x=-10 y=18 width=20 height=20" uuid="_seSHoCnAEeKD4vESWQFXyg" type="internal_type"/>
        </children>
```

```
       <children name="demuxServ:Demux" position="x=130 y=104 width=161 height=96" uuid="_fbGfACndEeKD4vESWQFXyg"
type="_DR_DwCnTEeKD4vESWQFXyg">
         <ports name="flowport5" position="x=-10 y=25 width=20 height=20" uuid="_kg8-gCndEeKD4vESWQFXyg" type="internal_type"/>
         <ports name="flowport6" position="x=151 y=-5 width=20 height=20" uuid="_lvJ3ACndEeKD4vESWQFXyg" type="internal_type"/>
         <ports name="flowport7" position="x=151 y=15 width=20 height=20" uuid="_mQtr8CndEeKD4vESWQFXyg" type="internal_type"/>
         <ports name="flowport8" position="x=151 y=35 width=20 height=20" uuid="_nu120CndEeKD4vESWQFXyg" type="internal_type"/>
         <ports name="flowport9" position="x=151 y=55 width=20 height=20" uuid="_pMjyACndEeKD4vESWQFXyg" type="internal_type"/>
         <ports name="flowport11" position="x=151 y=80 width=20 height=20" uuid="_JxpaQD7ZEeKtx69ukyHZJQ" type="internal_type"/>
       </children>
       <children name="motor:Servos" position="x=420 y=279 width=142 height=50" uuid="_DR1eMD7ZEeKtx69ukyHZJQ"
type="_ikEwkB30EeKNSu6wH1bjLA">
         <ports name="flowport7" position="x=-10 y=15 width=20 height=20" uuid="_IlphMD7ZEeKtx69ukyHZJQ" type="internal_type"/>
       </children>
     </children>
     <lines name="connector12" type="Connector" source_uml="_vffO0CnAEeKD4vESWQFXyg" destination_uml="_kg8-gCndEeKD4vESWQFXyg"/>
     <lines name="connector13" type="Connector" source_uml="_lvJ3ACndEeKD4vESWQFXyg" destination_uml="_seSHoCnAEeKD4vESWQFXyg"/>
     <lines name="connector14" type="Connector" source_uml="_mQtr8CndEeKD4vESWQFXyg" destination_uml="_dA2FoCByEeK_MPAvFvsPcA"/>
     <lines name="connector15" type="Connector" source_uml="_nu120CndEeKD4vESWQFXyg" destination_uml="_J-8IECgfEeKoD_SkLDcRiQ"/>
     <lines name="connector16" type="Connector" source_uml="_pMjyACndEeKD4vESWQFXyg" destination_uml="_dx5vICByEeK_MPAvFvsPcA"/>
     <lines name="connector17" type="Connector" source_uml="_JxpaQD7ZEeKtx69ukyHZJQ" destination_uml="_IlphMD7ZEeKtx69ukyHZJQ"/>
     <ports name="ServoData" position="x=-10 y=165 width=20 height=20" uuid="_vffO0CnAEeKD4vESWQFXyg" type="model_type"/>
   </parts>
  </sim:Model>
  <sim:Model name="D_Sensors_internal" simulinkName="Sensors/SysMLmodel/" type="InternalBlock">
    <parts name="Sensors" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=75 y=40 width=721 height=386" uuid="_kuL3sB30EeKNSu6wH1bjLA"
type="Class">
      <children name="compartment_sysml_structure">
        <children name="BPM085:Barometer" position="x=235 y=139 width=176 height=51" uuid="_tp_OsCBdEeK_MPAvFvsPcA"
type="_5TRf0B4JEeKNSu6wH1bjLA">
          <ports name="flowport1" position="x=166 y=15 width=20 height=20" uuid="_dHn_4CBuEeK_MPAvFvsPcA" type="internal_type"/>
        </children>
        <children name="part1:Batery" position="x=275 y=264 width=141 height=51" uuid="_ONQXECgpEeKoD_SkLDcRiQ"
type="_PLpR8CgpEeKoD_SkLDcRiQ">
          <ports name="flowport1" position="x=131 y=10 width=20 height=20" uuid="_RGktkCgpEeKoD_SkLDcRiQ" type="internal_type"/>
        </children>
        <children name="MT3329:GPS" position="x=285 y=19 width=125 height=53" uuid="_ve-3sDsHEeKxf-mhQd8wzw" type="_wFKt8DsHEeKxf-
mhQd8wzw">
          <ports name="flowport1" position="x=115 y=15 width=20 height=20" uuid="_yIpooDsHEeKxf-mhQd8wzw" type="internal_type"/>
        </children>
        <children name="MP6000:IMU" position="x=290 y=194 width=122 height=52" uuid="_9zCzMDsHEeKxf-mhQd8wzw" type="_MY8XgDsGEeKxf-
mhQd8wzw">
          <ports name="flowport2" position="x=112 y=19 width=20 height=20" uuid="_B0MUcDsIEeKxf-mhQd8wzw" type="internal_type"/>
        </children>
      </children>
      <lines name="connector18" type="Connector" source_uml="_RGktkCgpEeKoD_SkLDcRiQ" destination_uml="_ei4A8Cj1EeK74PyobivnqQ"/>
      <lines name="connector20" type="Connector" source_uml="_StnDICj1EeK74PyobivnqQ" destination_uml="_dOfn8Cj1EeK74PyobivnqQ"/>
      <lines name="connector21" type="Connector" source_uml="_dHn_4CBuEeK_MPAvFvsPcA" destination_uml="_dOfn8Cj1EeK74PyobivnqQ"/>
      <lines name="connector22" type="Connector" source_uml="_yIpooDsHEeKxf-mhQd8wzw" destination_uml="_DaaJgCduEeKH4dsWRuHDXw"/>
      <lines name="connector23" type="Connector" source_uml="_B0MUcDsIEeKxf-mhQd8wzw" destination_uml="_P5AP8Dl8EeKcJ4TEKT5BRw"/>
      <ports name="GPS Data" position="x=711 y=75 width=20 height=20" uuid="_DaaJgCduEeKH4dsWRuHDXw" type="model_type"/>
      <ports name="Pressure Data" position="x=711 y=146 width=20 height=20" uuid="_dOfn8Cj1EeK74PyobivnqQ" type="model_type"/>
      <ports name="Batery charge" position="x=711 y=320 width=20 height=20" uuid="_ei4A8Cj1EeK74PyobivnqQ" type="model_type"/>
      <ports name="IMU" position="x=711 y=255 width=20 height=20" uuid="_P5AP8Dl8EeKcJ4TEKT5BRw" type="model_type"/>
    </parts>
  </sim:Model>
  <sim:Model name="D_Payload_internal" simulinkName="Payload/SysMLmodel/" type="InternalBlock">
    <parts name="Payload" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=40 y=60 width=916 height=366" uuid="_QhWK8B30EeKNSu6wH1bjLA"
type="Class">
      <children name="compartment_sysml_structure">
        <children name="enableCamera:CameraActivator" position="x=420 y=159 width=186 height=51" uuid="_fzxZYCQZEeKPk4amLu4W5w"
type="_h8dkMCQZEeKPk4amLu4W5w">
          <ports name="flowport1" position="x=-10 y=15 width=20 height=20" uuid="_aMkq4CQaEeKPk4amLu4W5w" type="internal_type"/>
          <ports name="flowport2" position="x=176 y=15 width=20 height=20" uuid="_d-WTYCQaEeKPk4amLu4W5w" type="internal_type"/>
          <ports name="Enable" position="x=160 y=-10 width=20 height=20" uuid="_dZOjYCQ2EeKPk4amLu4W5w" type="internal_type"/>
        </children>
        <children name="video1:VideoDisplay" position="x=660 y=159 width=156 height=49" uuid="_tthXcCQZEeKPk4amLu4W5w"
type="_v7PzQCQZEeKPk4amLu4W5w">
          <ports name="flowport1" position="x=-10 y=16 width=20 height=20" uuid="_ZKzr4CQaEeKPk4amLu4W5w" type="internal_type"/>
        </children>
        <children name="RGB:Camera" position="x=175 y=124 width=176 height=51" uuid="_xgDbgCqIEeKGv79DEq1CwA"
type="_odpzEB30EeKNSu6wH1bjLA">
          <ports name="flowport4" position="x=166 y=22 width=20 height=20" uuid="_CMwpYCqJEeKGv79DEq1CwA" type="internal_type"/>
        </children>
      </children>
      <lines name="connector4" type="Connector" source_uml="_d-WTYCQaEeKPk4amLu4W5w" destination_uml="_ZKzr4CQaEeKPk4amLu4W5w"/>
      <lines name="connector6" type="Connector" source_uml="_QzBcICmtEeKD4vESWQFXyg" destination_uml="_dZOjYCQ2EeKPk4amLu4W5w"/>
      <lines name="connector7" type="Connector" source_uml="_CMwpYCqJEeKGv79DEq1CwA" destination_uml="_aMkq4CQaEeKPk4amLu4W5w"/>
      <lines name="connector8" type="Connector" source_uml="_C0f-0CqJEeKGv79DEq1CwA" destination_uml="_aMkq4CQaEeKPk4amLu4W5w"/>
      <lines name="connector9" type="Connector" source_uml="_d-WTYCQaEeKPk4amLu4W5w" destination_uml="_PL9xgDmBEeKcJ4TEKT5BRw"/>
      <ports name="payload_activator" position="x=-10 y=82 width=20 height=20" uuid="_QzBcICmtEeKD4vESWQFXyg" type="model_type"/>
      <ports name="Camera_OUT" position="x=906 y=113 width=20 height=20" uuid="_PL9xgDmBEeKcJ4TEKT5BRw" type="model_type"/>
    </parts>
  </sim:Model>
  <sim:Model name="ReqDiagram" simulinkName="SysMLmodel/" type="RequirementDiagram">
    <parts name="Barometer" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=600 y=370 width=106 height=31"
uuid="_5TRf0B4JEeKNSu6wH1bjLA" type="Class"/>
  </sim:Model>
  <sim:Model name="UseCaseDiagram" simulinkName="SysMLmodel/" type="UseCase"/>
  <sim:Model name="D_CameraActivator" simulinkName="CameraActivator/SysMLmodel/" type="InternalBlock">
    <parts name="CameraActivator" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=40 y=40 width=500 height=250"
uuid="_h8dkMCQZEeKPk4amLu4W5w" type="Class">
      <children name="compartment_sysml_structure"/>
      <lines name="connector1" type="Connector" source_uml="_aMkq4CQaEeKPk4amLu4W5w" destination_uml="_d-WTYCQaEeKPk4amLu4W5w"/>
      <ports name="flowport2" position="x=490 y=101 width=20 height=20" uuid="_d-WTYCQaEeKPk4amLu4W5w" type="model_type"/>
      <ports name="flowport1" position="x=-10 y=106 width=20 height=20" uuid="_aMkq4CQaEeKPk4amLu4W5w" type="model_type"/>
    </parts>
  </sim:Model>
  <sim:Model name="SM_FlightControl" simulinkName="Roll control/Roll Controller/SysMLmodel/" type="PapyrusUMLStateMachineDiagram">
```

```
    <parts name="Roll control" simulinkName="_t4ScQCEUEeKMCvIEM15I8A" position="x=30 y=30 width=991 height=446" uuid="_B1grQCm-
EeKD4vESWQFXyg" type="StateMachine">
      <children name="Roll" position="x=-10 y=105 width=20 height=20" uuid="_EpOJACm-EeKD4vESWQFXyg" type="Pseudostate:entryPoint"/>
      <children name="Aileron_R" position="x=976 y=80 width=20 height=20" uuid="_HO9cMCm-EeKD4vESWQFXyg" type="Pseudostate:exitPoint"/>
      <children name="Aileron_L" position="x=976 y=140 width=20 height=20" uuid="_F1eAECnGEeKD4vESWQFXyg" type="Pseudostate:exitPoint"/>
      <children name="Yaw" position="x=-10 y=313 width=20 height=20" uuid="_0UUWMD79EeKtx69ukyHZJQ" type="Pseudostate:entryPoint"/>
      <children name="Rudder" position="x=981 y=312 width=20 height=20" uuid="_1SJQYD79EeKtx69ukyHZJQ" type="Pseudostate:exitPoint"/>
      <children name="roll_controller" position="x=10 y=15 width=991 height=211" uuid="_DwOEcCm-EeKD4vESWQFXyg" type="Region">
        <children>
          <children name="Init \n  Aileron_L=0;Aileron_T-0;" position="x=115 y=6 width=201 height=41" uuid="_LeYqACm-EeKD4vESWQFXyg"
type="State"/>
          <children name="roll left \n Aileron_L=Aileron_L+1;Aileron_R=Aileron_R-1;" position="x=135 y=111 width=-1 height=-1"
uuid="_Qxe7UCm-EeKD4vESWQFXyg" type="State"/>
          <children name="roll_right \n Aileron_L=Aileron_L-1;Aileron_R=Aileron_R+1;" position="x=530 y=71 width=-1 height=-1"
uuid="_VhuoICnHEeKD4vESWQFXyg" type="State"/>
          <children name="ReadValue" position="x=390 y=6 width=73 height=50" uuid="_x8jhoCnVEeKD4vESWQFXyg" type="State"/>
          <children name="Initial0" position="x=40 y=11 width=-1 height=-1" uuid="_Ke7YICm-EeKD4vESWQFXyg" type="Pseudostate:initial"/>
        </children>
      </children>
      <children name="yaw_controller" position="x=10 y=226 width=991 height=220" uuid="_xWXXUD79EeKtx69ukyHZJQ" type="Region">
        <children>
          <children name="Init \n  Rudder=0;" position="x=95 y=36 width=102 height=42" uuid="_TuvOUD8BEeKtx69ukyHZJQ" type="State"/>
          <children name="ReadValue" position="x=338 y=39 width=78 height=38" uuid="_YmGl4D8BEeKtx69ukyHZJQ" type="State"/>
          <children name="yaw_left \n Rudder=Rudder-1;" position="x=257 y=130 width=-1 height=-1" uuid="_ZBlA8D8BEeKtx69ukyHZJQ"
type="State"/>
          <children name="yaw_right \n Rudder=Rudder+1;" position="x=533 y=105 width=-1 height=-1" uuid="_ZZOr8D8BEeKtx69ukyHZJQ"
type="State"/>
          <children name="Initial0" position="x=25 y=46 width=-1 height=-1" uuid="_RpD7QD8BEeKtx69ukyHZJQ" type="Pseudostate:initial"/>
        </children>
      </children>
      <lines name="Transition0" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@122908 (sourceX: 9, sourceY: 2,
targetX: -110, targetY: 0)" uuid="_QJLrkCnGEeKD4vESWQFXyg" type="Transition" source_uml="_Ke7YICm-EeKD4vESWQFXyg"
destination_uml="_LeYqACm-EeKD4vESWQFXyg"/>
      <lines name="Transition2" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@d0adf (sourceX: -3, sourceY: 21,
targetX: 16, targetY: -74)" uuid="_66Y_UCnVEeKD4vESWQFXyg" type="Transition" source_uml="_LeYqACm-EeKD4vESWQFXyg"
destination_uml="_x8jhoCnVEeKD4vESWQFXyg"/>
      <lines name="[(Roll&amp;lt;45 &amp;amp;&amp;amp; Aileron_R&amp;lt;Roll)]"
position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@1aaecc (sourceX: 27, sourceY: 25, targetX: -103, targetY: -26)"
uuid="_8yyrYCnVEeKD4vESWQFXyg" type="Transition" source_uml="_x8jhoCnVEeKD4vESWQFXyg" destination_uml="_VhuoICnHEeKD4vESWQFXyg"/>
      <lines name="[(Roll>-45 &amp;amp;&amp;amp; Aileron_R>Roll)]" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@fffe7dcf
(sourceX: -35, sourceY: 12, targetX: 286, targetY: -99)" uuid="_IFcZcCnYEeKD4vESWQFXyg" type="Transition"
source_uml="_x8jhoCnVEeKD4vESWQFXyg" destination_uml="_Qxe7UCm-EeKD4vESWQFXyg"/>
      <lines name="[after(0.01,sec)]" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@ffc87bdb (sourceX: -154,
sourceY: 20, targetX: 219, targetY: -25)" uuid="_PwR-MCnYEeKD4vESWQFXyg" type="Transition" source_uml="_VhuoICnHEeKD4vESWQFXyg"
destination_uml="_x8jhoCnVEeKD4vESWQFXyg"/>
      <lines name="[after(0.01,sec)]" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@1646e (sourceX: -28, sourceY:
-5, targetX: 214, targetY: 44)" uuid="_SwpX4CnYEeKD4vESWQFXyg" type="Transition" source_uml="_Qxe7UCm-EeKD4vESWQFXyg"
destination_uml="_x8jhoCnVEeKD4vESWQFXyg"/>
      <lines name="Transition0" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@1288d1 (sourceX: 10, sourceY: -5,
targetX: -63, targetY: 0)" uuid="_pUkHQD8BEeKtx69ukyHZJQ" type="Transition" source_uml="_RpD7QD8BEeKtx69ukyHZJQ"
destination_uml="_TuvOUD8BEeKtx69ukyHZJQ"/>
      <lines name="Transition1" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@2535b7 (sourceX: 51, sourceY: 1,
targetX: -169, targetY: -1)" uuid="_p0wDMD8BEeKtx69ukyHZJQ" type="Transition" source_uml="_TuvOUD8BEeKtx69ukyHZJQ"
destination_uml="_YmGl4D8BEeKtx69ukyHZJQ"/>
      <lines name="[(Yaw&amp;lt;45 &amp;amp;&amp;amp; Rudder&amp;lt;Yaw)]" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@18b8c7
(sourceX: 23, sourceY: 19, targetX: -218, targetY: -48)" uuid="_rCfBoD8BEeKtx69ukyHZJQ" type="Transition"
source_uml="_YmGl4D8BEeKtx69ukyHZJQ" destination_uml="_ZZOr8D8BEeKtx69ukyHZJQ"/>
      <lines name="[after(0.01,sec)]" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@fffbb61b (sourceX: -40,
sourceY: -20, targetX: 201, targetY: 47)" uuid="_rZsAsD8BEeKtx69ukyHZJQ" type="Transition" source_uml="_ZZOr8D8BEeKtx69ukyHZJQ"
destination_uml="_YmGl4D8BEeKtx69ukyHZJQ"/>
      <lines name="[(Yaw>-45 &amp;amp;&amp;amp; Rudder>Yaw)]" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@fffca693
(sourceX: -39, sourceY: 18, targetX: 52, targetY: -83)" uuid="_rwNDQD8BEeKtx69ukyHZJQ" type="Transition"
source_uml="_YmGl4D8BEeKtx69ukyHZJQ" destination_uml="_ZBlA8D8BEeKtx69ukyHZJQ"/>
      <lines name="[after(0.01,sec)]" position="org.eclipse.gmf.runtime.notation.datatype.RelativeBendpoint@175614 (sourceX: 21, sourceY:
-20, targetX: -20, targetY: 72)" uuid="_sEfRMD8BEeKtx69ukyHZJQ" type="Transition" source_uml="_ZBlA8D8BEeKtx69ukyHZJQ"
destination_uml="_YmGl4D8BEeKtx69ukyHZJQ"/>
    </parts>
  </sim:Model>
  <sim:Model name="D_Communication" simulinkName="Communication/SysMLmodel/" type="InternalBlock">
    <parts name="Communication" simulinkName="_47ibcB3sEeKGdNgBEZkxoQ" position="x=60 y=35 width=1101 height=446" uuid="_l-
y_MDl4EeKcJ4TEKT5BRw" type="Class">
      <children name="compartment_sysml_structure">
        <children name="Desired Waypoint:Constant" position="x=510 y=39 width=185 height=49" uuid="_FFp2QD1aEeKJ1dShKCcz2w"
type="_V5UicCQZEeKPk4amLu4W5w">
          <ports name="flowport7" position="x=175 y=20 width=20 height=20" uuid="_XN0VgD1aEeKJ1dShKCcz2w" type="internal_type"/>
        </children>
        <children name="Desired Altitude:Constant" position="x=515 y=94 width=183 height=50" uuid="_J3z9sD1aEeKJ1dShKCcz2w"
type="_V5UicCQZEeKPk4amLu4W5w">
          <ports name="flowport8" position="x=173 y=14 width=20 height=20" uuid="_X06xwD1aEeKJ1dShKCcz2w" type="internal_type"/>
        </children>
        <children name="Desired Pitch Angle:Constant" position="x=510 y=204 width=186 height=51" uuid="_Pz794D1aEeKJ1dShKCcz2w"
type="_V5UicCQZEeKPk4amLu4W5w">
          <ports name="flowport10" position="x=176 y=17 width=20 height=20" uuid="_Y9m4ID1aEeKJ1dShKCcz2w" type="internal_type"/>
        </children>
        <children name="Desired Airspeed:Constant" position="x=525 y=149 width=171 height=51" uuid="_UKIiED1aEeKJ1dShKCcz2w"
type="_V5UicCQZEeKPk4amLu4W5w">
          <ports name="flowport9" position="x=161 y=14 width=20 height=20" uuid="_YMe9MD1aEeKJ1dShKCcz2w" type="internal_type"/>
        </children>
        <children name="m1:Mux" position="x=790 y=124 width=102 height=78" uuid="_iDYw4D1aEeKJ1dShKCcz2w" type="_ixqWsCnTEeKD4vESWQFXyg">
          <ports name="flowport16" position="x=-10 y=-3 width=20 height=20" uuid="_mq8x4D1aEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport17" position="x=-10 y=17 width=20 height=20" uuid="_nWS08D1aEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport18" position="x=-10 y=37 width=20 height=20" uuid="_nxfjMD1aEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport19" position="x=-10 y=59 width=20 height=20" uuid="_o07WID1aEeKJ1dShKCcz2w" type="internal_type"/>
          <ports name="flowport20" position="x=92 y=25 width=20 height=20" uuid="_pQdbkD1aEeKJ1dShKCcz2w" type="internal_type"/>
        </children>
        <children name="receiver:XBee" position="x=175 y=94 width=136 height=51" uuid="_ZnrasD4cEeKs1LXrmbRGKw"
type="_adNUgD4cEeKs1LXrmbRGKw">
          <ports name="flowport1" position="x=126 y=13 width=20 height=20" uuid="_lR5TUD8SEeK4Z_eEFKYWaA" type="internal_type"/>
        </children>
```

```
      </children>
      <lines name="connector1" type="Connector" source_uml="_XN0VgD1aEeKJ1dShKCcz2w" destination_uml="_mq8x4D1aEeKJ1dShKCcz2w"/>
      <lines name="connector2" type="Connector" source_uml="_X06xwD1aEeKJ1dShKCcz2w" destination_uml="_nWS08D1aEeKJ1dShKCcz2w"/>
      <lines name="connector3" type="Connector" source_uml="_YMe9MD1aEeKJ1dShKCcz2w" destination_uml="_nxfjMD1aEeKJ1dShKCcz2w"/>
      <lines name="connector4" type="Connector" source_uml="_Y9m4ID1aEeKJ1dShKCcz2w" destination_uml="_o07WID1aEeKJ1dShKCcz2w"/>
      <lines name="connector5" type="Connector" source_uml="_pQdbkD1aEeKJ1dShKCcz2w" destination_uml="_w0lY0Dl_EeKcJ4TEKT5BRw"/>
      <ports name="Nav_cmd" position="x=-10 y=335 width=20 height=20" uuid="_u17TwDl_EeKcJ4TEKT5BRw" type="model_type"/>
      <ports name="Received Data" position="x=1091 y=193 width=20 height=20" uuid="_w0lY0Dl_EeKcJ4TEKT5BRw" type="model_type"/>
      <ports name="Payload" position="x=-10 y=370 width=20 height=20" uuid="_aD8AoDmBEeKcJ4TEKT5BRw" type="model_type"/>
    </parts>
  </sim:Model>
</xmi:XMI>
```