Modelagem de sistemas complexos usando SysML e OPM, adição de funcionalidades à ferramenta OPM e análise de usabilidade da mesma para modelagem de sistemas no LIT

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA
(PIBIC/CNPq/INPE)

Bruno Andrade Freitas Falcão (ITA, Bolsista PIBIC/CNPq)
E-mail: brunoaffalcao@gmail.com

Dr. Geilson Loureiro (LIT/INPE, Orientador)
E-mail: geilson@lit.inpe.br

COLABORADORES

Felipe Simon (ITA, ex-bolsista PIBIC/CNPq)

Julho de 2009

ÍNDICE

1 INTRODUÇÃO

O processo de concepção de sistemas complexos normalmente requer do modelador o trabalho com diversas alternativas de subsistemas para desempenho de atividades funcionais. A maioria das abordagens de engenharia de produto se utiliza de um desenvolvimento top-down, partindo da análise de requisitos para se abstrair funções que os supram e uma camada física que dê suporte às funções [5, Hatley & Pibhai, 1988]. As linguagens de modelagem de sistemas, além de suporte a todos os aspectos de produto (desempenhando o papel de ferramenta de manutenção) e interface de trabalho ágil, devem possuir uma apresentação que permita o gerenciamento de complexidade, permitindo-se uma fácil transição entre uma visão global e uma de subsistemas específicos, bem como garantir a reutilização dos modelos desenvolvidos na evolução dos sistemas [4, Rational Software Corp., 1988]. Sob a questão da capacidade de gerenciamento de complexidade abstrai-se a importância da elaboração de conceitos enxutos, que comuniquem efetivamente modeladores e *stakeholders*. Sob esta ótica foi desenvolvida a linguagem OPM [2, Dori, 2002], utilizada atualmente apenas no âmbito acadêmico na divisão de sistemas de engenharia do MIT, a qual fundamentalmente é diferenciada pela simplicidade de notação. Este trabalho visou a compreender pontos fortes desta linguagem ou potencialidades a serem desenvolvidas agregando-se algumas semânticas utilizando-se como parâmetro SysML [OMG, 2007], o padrão internacional estabelecido.

2 OBJETIVOS

Este projeto tem por objetivo enumerar as capacidades e limitações da ferramenta de modelagem de sistemas chamada OPM (Object-Process Methodology), desenvolvida pelo grupo do qual o orientador participou quando de seu estágio pós-doutoral no MIT (Massachusetts Institute of Technology), analisar a viabilidade da tradução entre esta e SysML (linguagem de padrão internacional) e identificar capacidades de cada uma das ferramentas de modelagem propondo adições de capacidades ao escopo de OPM que esta não possua e mapeando casos de aplicação de cada uma das linguagens.

Objetivos específicos:

1) Estudar OPM (Object Process Methodology). OPM é uma abordagem de desenvolvimento (evolução e manutenção) de sistemas baseada nas entidades "objeto", "processo" e "estado" e nas relações entre eles, incluindo todos em um único tipo de diagrama, o OPD (Object-Process Diagram), que apresenta um equivalente textual chamado OPL (Object-Process Language). Integra, portanto, em uma única representação características estruturais e comportamentais do sistema.
2) Estudar SysML (Systems Modeling Language). SysML é uma linguagem de modelagem de sistemas que foi adaptada em um esforço conjunto do INCOSE (International Council on Systems Engineering's) e do OMG (Object Management

Group) da linguagem de modelagem de software orientado a objeto UML 2.0. Possui notações específicas para análise de requisitos, análise funcional e modelagem de estrutura e comportamento da arquitetura de sistemas.

3) Estudar as leis de tradução entre OPM e SysML desenvolvidas no MIT pelo antigo aluno de iniciação científica, Felipe Simon, orientado pelo prof. Geilson.
4) Enumerar as principais capacidades de OPM e as principais limitações.
5) Enumerar as principais capacidades apresentadas por SysML e não por OPM.
6) Propor adições ao escopo de OPM das capacidades julgadas necessárias, dentre as identificadas em SysML.
7) Determinar e exemplificar categorias de soluções que OPM se mostra poderosa para abordar no desenvolvimento e manutenção de sistemas complexos e categorias de soluções que OPM se mostra deficiente para abordar.

3 PLANO DE TRABALHO

1) Revisão bibliográfica dos assuntos:
   1.1) OPM – linguagem de modelagem de sistemas que permite modelagem de estrutura e comportamento de sistemas usando uma mesma notação [Dori, 2002];
   1.2) SysML - Systems Modeling Language, desenvolvida a partir de UML [4], difere da mesma por conter permitir rastreabilidade de requisitos e modelagem de parâmetros físicos em seus modelos paramétricos.
   1.3) Total View Framework – framework desenvolvido pelo prof. Geilson Loureiro como uma abordagem de modelagem de requisitos, funcional e física de produto, seus processos de ciclo de vida e as organizações responsáveis por esses processos. [Loureiro, 1999]
2) Aprendizado e utilização de ferramentas de modelagem computacionais disponíveis:
   2.1) OPCaT (Object-Process Case Tool) - ferramenta computacional que permite a elaboração de modelos com notação OPM. [www.opcat.com]
   2.2) Rhapsody - ferramenta computacional que permite a elaboração de modelos com notação SysML. [Artisan, 2009]
3) Aprendizado prático de modelagem de sistemas, identificando forças e limitações de cada uma
   3.1) Modelar sistemas-exemplo presentes em publicações já realizadas como tutorial e novos sistemas simples para praticar o uso das ferramentas
   3.2) Estudar as leis de tradução entre OPM e SysML
   3.3) Praticar a tradução dos sistemas modelados anteriormente de OPM pra SysML e vice-versa
4) Desenvolvimento de modelos de um produto em SysML e tradução do mesmo para OPM e desenvolvimento em OPM e tradução para SysML, envolvendo:

4.1) Modelos de requisitos - análise dos stakeholders, das medidas de efetividade e documentação de seus requisitos;

4.2) Modelos funcionais - contexto, interfaces externas, funções, análise de risco, estrutura funcional (funções e fluxos de energia, material e/ou informação) e comportamento funcional (estados, transições, ações, habilitação e desabilitação de funções);

4.3) Modelagem da arquitetura física - contexto, interfaces externas, análise de risco, arquitetura de fluxos, arquitetura de interconexões;

4.4) Modelos de atributos - parâmetros físicos e seus relacionamentos com desempenho, custo, risco e prazo de desenvolvimento;

4.5) Tradução do modelo.

5) Descrição das forças e limitações de OPM

5.1) Enumeração da semântica dos elementos de OPM e suas principais capacidades.

5.2) Enumeração das semânticas dos elementos de SysML ausentes em OPM e as principais limitações de OPM por não apresentar estas semânticas.

5.3) Seleção de semânticas necessárias a OPM.

6) Adicionar funcionalidades necessárias ao OPM

6.1) Criar novos elementos ou adicionar características aos elementos de OPM que adicionem estas semânticas. Criar seu equivalente em OPL.

7) Conclusão: Avaliar a usabilidade de OPM

7.1) Determinar qualitativamente categorias de soluções que OPM se mostra poderosa ou deficiente para abordar no desenvolvimento e manutenção de sistemas complexos.

8) Documentação do projeto.

4 ANÁLISES INICIAIS

As etapas iniciais do trabalho consistiram na revisão bibliográfica proposta para as duas linguagens a serem dominadas, a partir das quais foram confeccionadas apresentações explanadoras de suas sintaxes e aplicações. A partir deste estudo foi possível a elaboração de modelos simples para familiarização tanto com as estruturas das linguagens, como com as ferramentas computacionais apresentadas no plano de trabalho.

Uma análise semântica comparativa entre SysML e OPM mais completa requereria, no entanto, um desenvolvimento completo de produto que evidenciasse as maiores necessidades do processo de modelagem diante de toda a simbologia fornecida por SysML, que neste contexto é utilizada como referência de abrangência de semânticas a serem cobertas.

Para desempenho de tal experiência foi escolhida a implementação do Total View Framework, que se trata de uma metodologia de desenvolvimento simultâneo de modelos

de requisitos, funcional e físico para produto, para os processos que ocorrem em seu ciclo de vida e para as organizações que realizam estes processos. Este framework foi desenvolvido pelo orientador em sua tese de doutorado e utiliza para notação diagramas de Análise Estruturada [6, Yourdon, 1989] (Data Flow Diagram, Entity Relationship Diagram e State Transition diagrams) e de Orientação a Objeto (UML) ou extensões destes.

A escolha de SysML como referência se fundamenta principalmente na sua adoção como linguagem internacional e por este ter sido desenvolvido com fins de modelagem de sistemas, agregando estruturas diferentes de UML 2.0 (de quem herda diretamente) como fluxo não só de dados, como de material e energia, e conexões físicas. Elementos que são necessários para modelagem de sistemas técnicos, segundo a formulação de Hubka & Eder [7]. Além disso, SysML tem foco na abordagem de todas as possibilidades de sistemas, possuindo muitas semânticas passíveis de um estudo para simplificação.

O desenvolvimento a partir do qual foram feitas as conclusões deste relatório baseiam-se no capítulo 7 da referência [3], que trata de modelagem de produto, processo e organização através de análise estruturada simultânea.

5 PROGRESSO

O trabalho investigou as características de cada uma das linguagens e da abordagem utilizada. Dentro destes aspectos são enumeradas algumas conclusões.

5.1 SysML

A linguagem procura uma bem determinada separação para representação de aspectos estruturais e comportamentais do sistema de forma a existir complementaridade. A estrutura completa é composta de 11 diagramas que envolvem mais de uma centena de símbolos e marcações para os objetos (Stereotypes). Dentre os diagramas 4 envolvem aspectos estruturais, 4 comportamentais e 3 são chamados de Cross-Cutting, inseridos para dar suporte à análise de requisitos, à rastreabilidade entre diagramas e à estrutura de pacotes que confere a característica de meta-linguagem. Esta estrutura muito descritiva pode atrapalhar a etapa de concepção em que a preocupação com que simbologia utilizar pode interferir na forma adequada de descrever o sistema.

5.2 OPM

Modela sistemas através de 3 entidades: "objetos", "processos" e "estados" e das relações entre eles por 18 links que podem ter natureza estrutural ou procedural. Desta forma, agrega-se em um único diagrama, o chamado OPD (Object-Process Diagram), aspectos estruturais e comportamentais do sistema. Há um equivalente textual do diagrama,

chamado OPL (Object-Process Language) que apresenta uma estrutura lingüística para cada relacionamento no OPD. Esta integração, ao mesmo tempo que fornece uma visão global do sistema pode levar facilmente à desorganização e deixar o diagrama ilegível para modelos robustos e com longa manutenção ou longos processos iterativos de modelagem.

5.3 Total View Framework

As etapas deste framework consistem basicamente de um diagrama de contexto em que o objeto final, seja o produto ou a organização, é mostrado ao centro do diagrama circulado por *stakeholders* ligados a este por links de relacionamentos marcados com os interesses de cada um. Neste desenvolvimento foi eliminada a seção correspondente a processos de ciclo de vida presente na referência [3], por ser feita a consideração de esta estar incluída nas funções de produto e de organização. A partir destes diagramas de contexto são derivadas as funções que serão localizadas em uma linha de tempo ou segundo estados dos mecanismos. Gerando-se simultaneamente a estrutura que desempenhará cada função determinada. Esta é representada em termos de componentes e dos fluxos de dados, material e energia entre estes.

Dentro deste escopo foram estruturados diagramas de SysML correspondentes a cada etapa como equivalentes da notação em análise estruturada utilizada originalmente. Estes estão denotados na tabela 5.1.

Tabela 5.1: Notações de SysML propostas em análise do Total View Framework.

	Produto	**Organização**
Requisitos	Block diagram Parametric diagram Requirements diagram	Parametric diagram Use Cases diagram (todos os cenários possíveis) + Requirements diagram
Funcional	Block diagram Activity + Sequence diagram + State machine diagram	Class diagram Use Cases diagram (cenários específicos) +Activity diagram
Física	Block diagram State machine diagram	Class diagram Sequence diagram

Cada combinação de diagramas visa a abranger aspectos estruturais e comportamentais de cada aspecto analisado, correspondente às células da tabela. Quando ambos referem-se ao mesmo aspecto estão indicados pelo sinal de "+" entre si. Neste caso os diagramas de blocos e paramétricos apresentam aspectos apenas de natureza estrutural e os diagramas de seqüência, atividade, casos de uso e estado de máquina apresentam aspectos apenas comportamentais.

Em SysML ainda é utilizado uma extensão do Requirement diagram, que utiliza blocos com o stereotype de <<Requirement>>, estes, em sua descrição, apresentam o *stakeholder* interessado no requisito. Estes diagramas são ditos de Cross-Cutting, ou seja, apresentam elementos pertencentes aos dois aspectos: estrutural e comportamental.

A equivalência destes em OPM trata-se de um OPD adequado. As figuras 5.3.1 e 5.3.2 apresentam um modelo diagrama de contexto para o levantamento de requisitos do produto "sistema de satélite". No apêndice são mostradas as leis de tradução mapeadas entre OPM e SysML desenvolvidas pelo aluno de iniciação anterior Felipe Simon (iniciação científica do INPE de agosto de 2006 a julho de 2007) em seu estágio curricular no MIT, bem como um outro modelo traduzido utilizando estas leis [8].

Em OPM o diagrama apresenta o objeto principal "Sistema Satélite" que apresenta como processos atributos os requisitos dos *stakeholders*, os quais neste caso são apresentados como objetos ambientais que são agentes destes processos-atributos. Esta se trata de uma adaptação da linguagem OPM para a análise de requisitos, que deve em seqüência ser realizada para análise funcional. Pode-se ver que a descrição realizada em 2 ou 3 diagramas SysML foi reduzida a um unido OPD, no entanto podem surgir penalidades como o acréscimo da quantidade de símbolos para se gerar a semântica desejada.

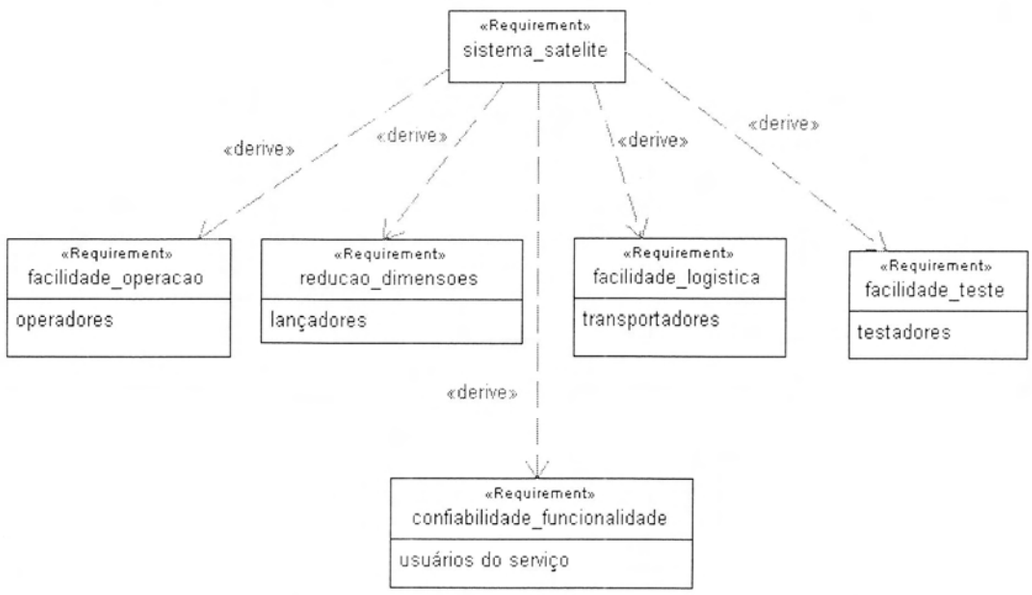

Figura 5.3.1: Diagrama de contexto com os requisitos de produto em um Requirements diagram em SysML.

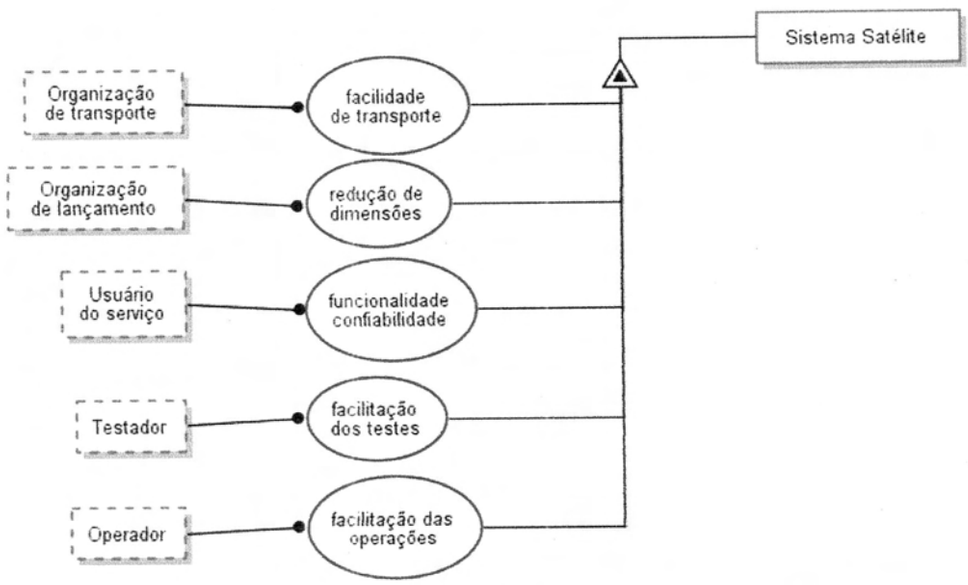

Figura 5.3.2: Diagrama de contexto com os requisitos de produto em OPM.

Neste ponto já é razoável concluir que, apesar de os diagramas mais adequados para análise de requisitos em SysML se modificarem para modelagem de organização, em OPM os diagramas não se modificarão drasticamente. Ou seja, para fornecer ao modelo as informações de organização foi utilizado o diagrama de casos de uso, o que reflete uma ênfase maior em comportamento do objeto estudado, ao passo que a ênfase em produto se deu na estrutura. Este comportamento, no entanto, tem forte natureza lógica e fraca natureza temporal ou de causalidade. Os fluxos envolvidos são principalmente de dados. Portanto, por mais que se modifique a abordagem SysML, o escopo em OPM não se altera muito.

Essa mudança de diagrama em SysML para desempenhar o mesmo papel (análise de requisitos) se dá por os requisitos em processos e organização, caso fossem expostos em Requirement diagram seriam colocados na forma "atender a tal circunstância de utilização do produto". Nesta situação, optou-se por utilizar um diagrama de casos de uso.

A partir do momento em que os diagramas usam parâmetros no tempo ou de causalidade, como em diagramas de seqüência, que utilizam linhas de ações e de vida, ou as linhas de sincronização em diagramas de atividade, os modelos passam a não ter tradução adequada sem adições textuais em OPM, que não dá suporte a esse tipo de semântica. Este tipo de encadeamento em OPM só pode ser tratado como fluxo de controle, tendo que ser construído um objeto-atributo "tempo" que utilizaria seu estado para controle de determinado processo. Pode-se dizer que esta estrutura agrega complexidade ao modelo, exatamente por falta de notação adequada.

No entanto, mesmo o fluxo de controle apresenta deficiências ainda em OPM, que não possui uma estrutura para vínculos paramétricos entre suas variáveis de controle. Não é possível distinguir um vínculo de um valor de referência sem que isso seja inserido textualmente. Esta falta de sintaxe prejudica o processo de automatização da tradução.

Tratando-se de aspectos de causalidade, um diagrama OPM não reflete com clareza em que ponto a atividade de seus componentes se inicia, a não ser pela indicação de estados iniciais dos objetos. Mas os aspectos lógicos que seguem à causalidade inicial são bem cobertos. Dessa forma, OPM mostra maior potencial para aplicação em comportamentos que não tenham forte dependência do tempo em si, mas sim de situações de controle (ativação pelo surgimento de um objeto ou por mudança de estado de um objeto), como é o caso dos diagramas de estado de máquina.

Por fim vê-se que estes 7 diagramas, 2 de estrutura: blocos e paramétricos; 4 comportamentais: casos de uso, seqüência, atividades e estado de máquina; e o diagrama de requisitos são os utilizados para a modelagem completa proposta pelo framework. A abrangência da análise apenas destes, de fato, evidencia as deficiências ou potencialidades de OPM. Em resumo, tem-se potencialidade para modelagem de estrutura e dos envolvimentos lógicos entre os componentes estruturais, abordando bem o diagrama de estado de máquina e alguns aspectos de atividades, ademais os diagramas de estruturas (blocos e paramétricos) e os modelos de requisitos (contexto), e deficiência para modelagens que envolvem linhas de tempo e causalidade fortemente dependente de condições iniciais, como nos casos dos diagramas de seqüência e novamente alguns aspectos de atividades.

Seria aconselhável, nesse sentido, a adição de semânticas envolvendo posicionamento no tempo e clareza de estados iniciais. Como entidades de linhas de tempo e fronteiras delimitando situações iniciais.

5.4 OPM como uma ferramenta de brainstorming e SysML como ferramenta de documentação

A complementaridade entre SysML e OPM torna-se, assim, evidente. OPM, neste sentido, por possuir pouca notação, agrega muita potencialidade em facilitar a construção de diagramas de contexto e etapas de concepção de um modelo, pois pouca notação atribui uma qualidade necessária aos dois processos: visão global do sistema. Uma notação carregada, em geral, conduz o modelador a, preocupando-se em adequar as estruturas corretas, partir para um sistema específico, visando a modelar mantendo coesão e coerência entre estrutura e comportamento. Isto significa queimar etapas do processo top-down de concepção de sistemas.

Este problema é recorrente em SysML. Quando o passo-a-passo de modelagem não é muito bem fechado, como no framework utilizado, tende-se a, após modelarem-se os casos de uso, partir-se para estrutura e comportamento na linha de tempo, devido a forte necessidade de coesão amarrar muito o sistema.

No entanto, esta coesão é necessária em etapas de manutenção do modelo. E nesta situação SysML mostra maior potencialidade. Quando a situação envolve evolução de sistemas essa utilização integrada se faz mais importante, o confronto entre o sistema documentado e o esboço da inovação.

Desta forma, é natural que se alie as qualidades das duas linguagens nesta utilização integrada. Mapeados os formatos de tradução para uma realização automática, pode-se trabalhar em OPM a nível de concepção e partir de um modelo traduzido para SysML na fase de desenvolvimento final e manutenção, utilizando-se um mecanismo iterativo para evolução.

6 TRABALHOS FUTUROS

Este trabalho tratou da eficiência mostrada pelas duas formas de desenvolvimento em diferentes etapas do processo de modelagem chegando-se à proposta de uso integrado nos ambientes em que se adequarem mais.

Como conseqüência natural da maior integração entre as duas linguagens sugere-se o trabalho na automatização deste processo de modo a se gerar diagramas de SysML a partir dos OPD's gerados na concepção aonde estes se apresentaram sem ambigüidade semântica.

Ainda neste sentido de automatização propõe-se o estudo da comunicação dos modelos gerados com softwares de simulação, como MATLAB ou outros, para validação da estrutura montada a partir de parâmetros computáveis.

Por fim, pode-se ainda estender o trabalho feito com o Total View Framework [3] para montagem efetiva de cada diagrama adequadamente para concepção completa do produto e organização.

7 CONCLUSÕES

Neste trabalho foi apresentada uma análise acerca dos mecanismos de tradução entre SysML e OPM e a possibilidade de acréscimo de semânticas a OPM tornando-a mais abrangente,conferindo-lhe capacidades como: melhor desenvolvimento de fluxos de controle em linha de tempo e capacidade de modelar vínculos entre estes. Esta, no entanto, já apresenta casos de melhor aplicação, como nos diagramas de contexto mostrados, por ser menos carregada de notação e estas situações serem independentes de tempo. A aplicabilidade deste estudo se dá na concepção mais ágil de sistemas favorecendo o approach top-down de maneira correta integrando-se as potencialidades das duas linguagens: OPM na concepção e SysML em manutenção e documentação de modelos. Este processo pode, portanto, ser feito de maneira automatizada para que não sejam necessários grandes esforços concebendo-se sistemas em SysML ou efetuando-se a tradução a partir de OPM, nem precisando interpretar os diagramas OPM que englobam muitos aspectos juntos, podendo tornar-se ilegíveis, na manutenção do sistema.

8 REFERÊNCIAS BIBLIOGRÁFICAS

[1] **SysML partners** (www.sysml.org <http://www.sysml.org>), **2005**. *Systems Modeling Language (SysML) specification*, version 1.0 alpha. Acessado em: 7 de outubro de 2008.

[2] **Dori, D. 2002** *Object-Process Methodology - A Holistic Systems Paradigm*. 1st ed., Springer-Verlag, New York, 2002. ISBN: 3-540-65471-2.

[3] **Loureiro**, G. 1999. *A systems engineering and concurrent engineering framework for the integrated development of complex products*. PhD Thesis, Loughborough University, Loughborough, UK.

[4] **Rational Software Corp. , 1998.** *Rational objectory process: your UML process.*

[5] **Hatley, D.J. & Pirbhai, I. A., 1988.** *Strategies for real-time system specification.* Dorset House Publishing, New York. ISBN:0-932633-11-0

[6] **Yourdon, E., 1989.** *Modern structured analysis.* Yourdon Press, Englewood Cliffs, New Jersey. **1989.** ISBN: 0-13598624-9.

[7] **Hubka, V. & Eder, W. E. , 1988.** *Theory of technical systems: a total concept theory for engineering design.* Springer-Verlag, Berlin. ISBN: 3-540-17451-6.

[8] **Simon, Felipe, 2008.** Relatório de estágio curricular – MIT.

[9] NASA Systems Engineering Handbook, SP-610S – June 1995

[10] K. Forsberg, H. Mooz, Systems engineering for faster,cheaper, better, Proceedings ofthe INCOSE'99, INCOSE,Brighton, 1999.

[11] D. Dori, I. Reinhartz-Berger, and A. Sturm, "Developing Complex Systems with Object-Process Methodology with OPCAT", presented at Conceptual Modeling - ER 2003, 2003.

[12] M. Hause, F. Thom, and Alan Moore, "Inside SysML", IEE Electronics Systems and Software – June/July 2005.

[13] http://ocw.mit.edu/OcwWeb/Engineering-Systems-Division/ESD-34January--IAP--2007/CourseHome/index.htm. Acessado em: 14 de setembro de 2008.

[14] https://www.opcat.com. Acessado em: 10 de agosto de 2008.

9 APÊNDICE

As tabelas de tradução presentes nas seções 9.4 e 9.5 foram desenvolvidas por Felipe Simon quando em seu estágio curricular no MIT e foram oferecidas para realização deste trabalho. As tabelas da seção 9.1, 9.2 e 9.3 apresentam as classificações dentro de cada linguagem a serem utilizadas nas seções seguintes para mapear a tradução.

9.1 Modelo estrutural

Glossary

Association: structural element that classifies physical links between blocks
Behavior: an executable preprogrammed sequence of changes on system's states
Connector: a physical link between blocks
Constraint Property: an expression owned by a block which represents a contraint to its properties
Flow: structural concept that represents a path where a flow can occur
Flow Property: a property owned by a block or interface that represents a flow
Navigable End: on a relationship between classifiers, a realtionship's navigable end indicates that the classifier connected to that end is visible by the other others
Operation: a function representation in SysML – it is usually a reference to some behavior starting point
Reference: a physical direct link to a block or some other classifier
Semantic Dependency: a relationship between two blocks where the definition of one is not complete without the other
User Defined Literal: A value that can be assumed by a data type which is represented by a sequence of characters

Table 9.1.1.: Classifiers - model elements that represent a set of objects on the real world

Nature	Physical, Data
Contains Operations	True, False
Contains Behaviors	True, False
Role	Instance, Classifier
Abstract	True, False, Any
External	True, False
Restriction	None, Only Flow Properties, Only Constraint Properties, Has Unit and Dimension, Is Composed of User Defined Literals

Table 9.1.2.: Relationships - model elements that represent how classifiers relate to each other

Number of Navigable ends	All, One, None
Nature	Whole/Part, Inheritance, Semantic Dependency, Flow, Other
Is Direct Physical Link	True, False
Is Also a Classifier	True, False
Links the Parts' Lifetimes	True, False

Table 9.1.3.: Block Attributes - features that can be added to the Block element

Nature	Physical, Data, Functional
Type	Service, Flow, Reference, Property, Constraint, Operation

Table 9.1.4.: Other Structural Features - features that can be added to other Classifiers

Nature	Physical, Data
Derived	True, False
Owner	FlowSpecification/FlowPort, Connector, Association, Operation, Enumeration

9.2 Modelo comportamental

Glossary

Action: smallest behavior executable parcticle
Namespace: container where the modeler can define new elements
Structured (Activity Group): a portion of the activity where the execution of any embedded actions may not begin until the structured activity node has received its object and control tokens and the availability of output tokens from the structured activity node does not occur until all embedded actions have completed execution
Expansion (Structured Activity Node): An expansion region is a strictly nested region of an activity with explicit input and outputs (modeled as ExpansionNodes). Each input is a collection of values. The expansion region is executed once for each element (or position) in the input collection.
Edge (Token Handling): an initial or final node for the token flow in a behavior
Interruptible (Activity Group): a region that when a token leaves it via edges designated by the region as interrupting edges, which may be activated by external events, all tokens and behaviors in the region are terminated
Lifeline: An individual participant in an interaction
Message (Interaction Fragment): specific kind of communication between two lifelines or between a lifeline and an unknown element in an Interaction.
Link: A temporary or permanent physical means of interaction between two blocks
Structural Feature: a typed feature of a classifier that specifies the structure of instances of the classifier

Table 9.2.1.: Behavior Elements - A top-level classification of the behavior diagram elements

Owner	Activity, Interaction, StateMachine
Constains Action	True, False
Contains Sub Behavior	True, False, Any
Contains Block or Data	True, False
Specifies Token Flow	True, False, Any
Is Sub Namespace	True, False

Table 9.2.1.1.: Activity Groups - Behavior Elements with values:

(Owner=Activity, Action=False, SubBehavior=True, BlockData=False, TokenFlow=False, Namespace=True)

Is Interruptible	True, False
Is Structured	True, False

Table 9.2.1.1.1.: Structured Activity Nodes - Activity Group elements with values:

(Interruptible=False, Structured=True)

Kind	Condition, Expansion, Loop, Sequence

Table 9.2.1.2.: Control Nodes - Behavior Elements with values:

(Owner=Activity, Action=False, SubBehavior=False, BlockData=False, TokenFlow=True, Namespace=False)

Position	Token Arrival, Token Departure
Token Handling	Sequencial, Parallel, Edge

Table 9.2.1.3.: Interaction Fragments: Behavior Elements with values:

(Owner=Interaction, Action=False, SubBehavior=True, BlockData=False, TokenFlow=Any, Namespace=True)

Nature	Action, Message, Sub Behavior, Token Flow Control, Constraint

Table 9.2.1.4.: Objects in Behaviors: Behavior Elements with value (BlockData=True)

Source	Internal, External
Owner	Behavior Itself, Behavior's Element
Number	Single, Multiple

Table 9.2.1.5. Action - A feature found in Behavior Elements with value (Action=True)

Nature	Accept Stimulus, Send Stimulus, Read, Write, Create, Destroy
Location	Internal,External
Target	Signal, Object, Link, Structural Feature, Variable, Test

9.3 OPM semantic dimensions

Table 9.3.1.: Things

Nature	Object, Process, State

Table 9.3.2.: Relations

Nature	Whole/Part, Inheritance, Exhibition, Instance/Classifier, Other One-Way, Other Both-Ways

Table 9.3.3.: Links: add functional information to a process

Activation Moment	Before Process Start		After Process Start	
	Object is Affected	True,False	Normal Course	True, False, Any
	Object is Trigger	True, False	Result	Affects, Yields

Object is Agent	True, False	Product	Object,	Process
Object is Event	True, False			

Table 9.3.4.: Multiple Token Flow

Selection	And, Or

9.4 Informação traduzível de SyML para OPM

Tabela 9.4.1: Modelo estrutural

Classifier()	Thing(Nature=Object)
Classifier(Role=Instance)	(from an **Object**) Relation(Nature=Instantiation) + Thing(Nature=Object)
Relationship(Nature=Whole/Part)	Relation(Nature=Whole/Part)
Relationship(Nature=Inheritance)	Relation(Nature=Inheritance)
Relationship(Nature=Other, Navigable Ends=All)	Relation(Nature=Other Both Ways)
Relationship(Nature=Other, Navigable Ends=One)	Relation(Nature=Other One Way)
Block Attribute(Nature=Port)	(from an **Object**) Relation(Nature=Exhibition) + Thing(Nature=Process) (Object that represents the block that owns the port is zoomable)
Block Attribute(Nature=Flow)	(from an **Object**) Relation(Nature=Exhibition) + Thing(Nature=Process) + Link(ActivationMoment=AfterProcessStart,Result= Yields, Product=Object)
Block Attribute(Nature=Property)	(from an **Object**) Relation(Nature=Exhibition) + Thing(Nature=Object)
Block Attribute(Nature=Operation)	(from an **Object**) Relation(Nature=Exhibition) + Thing(Nature=Process)
Feature(Owner=Operation)	(from an **Process**) Link(ActivationMoment=AfterProcessStart, Result=Affects, Product=Object) + Thing(Nature=Object)
Feature(Owner=Enumeration)	Thing(Nature=State)

Tabela 9.4.2: Modelo estrutural.

Element(ConstainsAction=True)	Thing(Nature=Process)
Element(ConstainsSubBehavior=True)	Thing(Nature=Process) (zoomable)
Element(ConstainsBlockData=True)	Thing(Nature=Object)
Element(IsSubNamespace)	Thing() (zoomable)
Element(	(from **Object**) Relation(Nature=Exhibition) +

SpecifiesTokenFlow=True/Any, Owner=Interaction, Nature=Message)	Thing(Nature=Process) + Link(ActivationMoment=AfterProcessStart, Result=Yields, Product=Process)
Element(SpecifiesTokenFlow=True, Owner=StateMachine)	(from **State** to **State**) Link(ActivationMoment=BeforeProcessStarts, ObjectAffected=True) + Thing(Nature=Process) + Link(ActivationMoment=AfterProcessStart,Result=Yields, Product=Object)
Control Node (Position=Departure, TokenHandling=Sequential)	Multiple Token Flow(Selection=or)
Control Node(Position=Departure, TokenHandling=Parallel)	Multiple Token Flow(Selection=and)
Interaction Fragment(Nature=Action)	Thing(Nature=Process)
Interaction Fragment(Nature=Message)	Thing(Nature=Process)
Interaction Fragment(Nature=Message)	Thing(Nature=Process) (zoomable)
Interaction Fragment(Nature=Message)	Link(ActivationMoment=AfterProcessStart,Result=Yields, Product=Process)

Tabela 9.4.3: Configurações de característica de elementos de OPM.

Classifier(External=True)	Thing(Nature=Object) (+ OPL text **is environmental**)
Object in Behavior (Source=External)	Thing(Nature=Object) (+ OPL text **is environmental**)

9.5 Informação traduzível de OPM para SysML

Tabela 9.5.1: Coisas.

Thing(Nature=Object)	Classifier
Thing(Nature=Process)	Behavior
Thing(Nature=State)	Behavior Element (Owner=StateMachine, TokenFlow=False)

Tabela 9.5.2: Relacionamentos.

Relationship(Nature=Whole/Part)	Relationship(Nature=Whole/Part)
Relationship(Nature=Inheritance)	Relationship(Nature=Inheritance)
Relationship(Nature=Exhibition) + Thing(Nature=Object)	Block Attribute(Nature=Data, Type=Attribute)
Relationship(Nature=Exhibition) + Thing(Nature=Process)	Block Attribute(Nature=Functional, Type=Operation)
Relationship(Nature=Instance/Classifier)+ Thing(Nature=Object)	Classifier(Role=Instance)
Relationship(Nature=Other One-Way)	Relationship(Nature=Other, NavigableEnd=One)
Relationship(Nature=Other Both-Ways)	Relationship(Nature=Other, NavigableEnd=All)

Tabela 9.5.3: Links.

Link(AtivationMoment=Before, Affected=True, Trigger=True, Agent=False, Event=False)	Feature(Derived=False, Owner=Operation) Object in Behavior(Number=Single)
Link(AtivationMoment=Before Affected=True, Trigger=True, Agent=False, Event=True)	Feature(Derived=False, Owner=Operation) Object in Behavior(Number=Single, Source=External)
Link(AtivationMoment=Before Affected=False, Trigger=True, Agent=False, Event=False)	Feature(Derived=False, Owner=Operation) Object in Behavior(Number=Single)
Link(AtivationMoment=Before Affected=False, Trigger=True, Agent=False, Event=True)	Feature(Derived=False, Owner=Operation) Object in Behavior(Number=Single, Source=External)
Link(AtivationMoment=Before Affected=False, Trigger=False, Agent=False, Event=False)	Control Node(Position=Token Arrival)
Link(AtivationMoment=Before Affected=False, Trigger=True, Agent=True, Event=False)	Classifier(Nature=Physical, External=True)
Link(AtivationMoment=After, NormalCourse=Any, Result=Affects, Product=Object)	Feature(Derived=False, Owner=Operation) Object in Behavior(Number=Single)
Link(AtivationMoment=After, NormalCourse=True, Result=Yields, Product=Process Execution)	Behavior Element(TokenFlow=True)
Link(AtivationMoment=After, NormalCourse=False, Result=Yields, Product=Process Execution)	Behavior Element(TokenFlow=True) Object in Behavior(Source=Internal,Number=Single)

9.6 Tradução do modelo da máquina de Turing.

Pode-se ver que a idéia de loop e aspectos semânticos temporais não são bem ilustrados pelo diagrama em OPM.

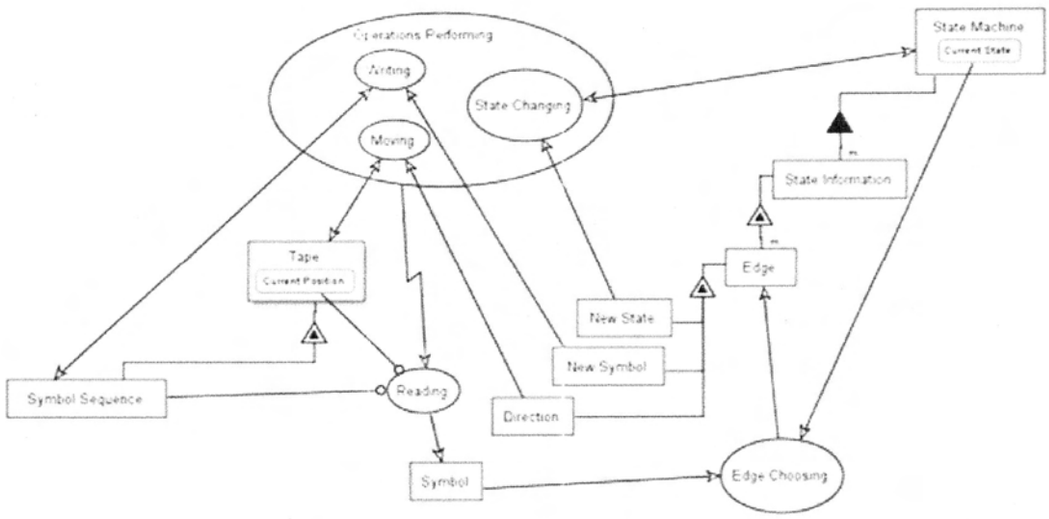

Figure 9.6.1: Turing Machine Model in OPM

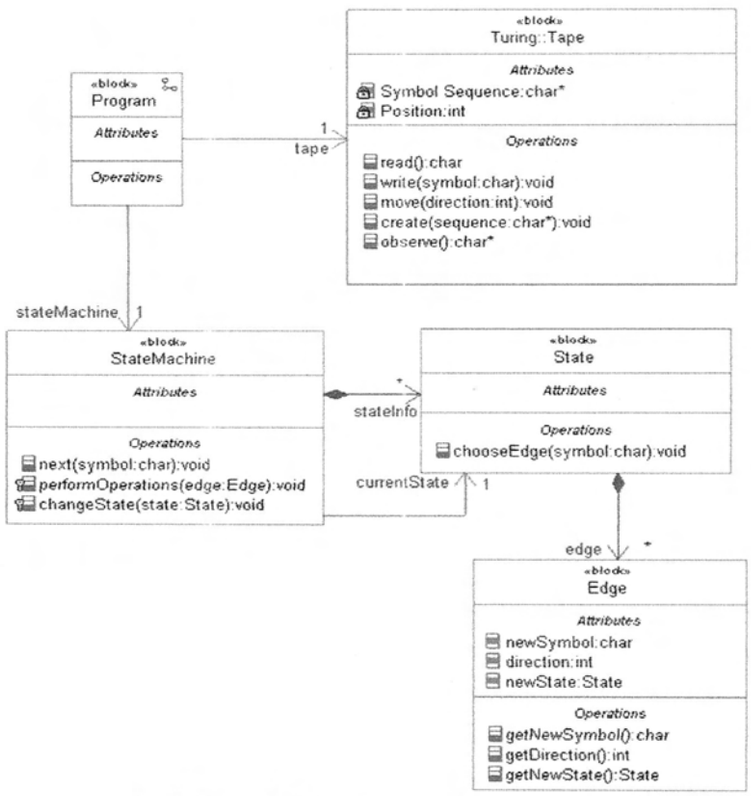

Figure 9.6.2: Turing Machine Structural Model in SysML

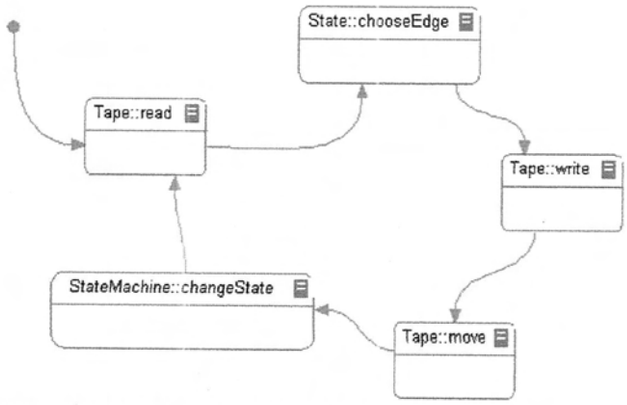

Figure 9.6.3: Turing Machine Behavioral Model in SysML Option 1 (Plain Activity Diagram).

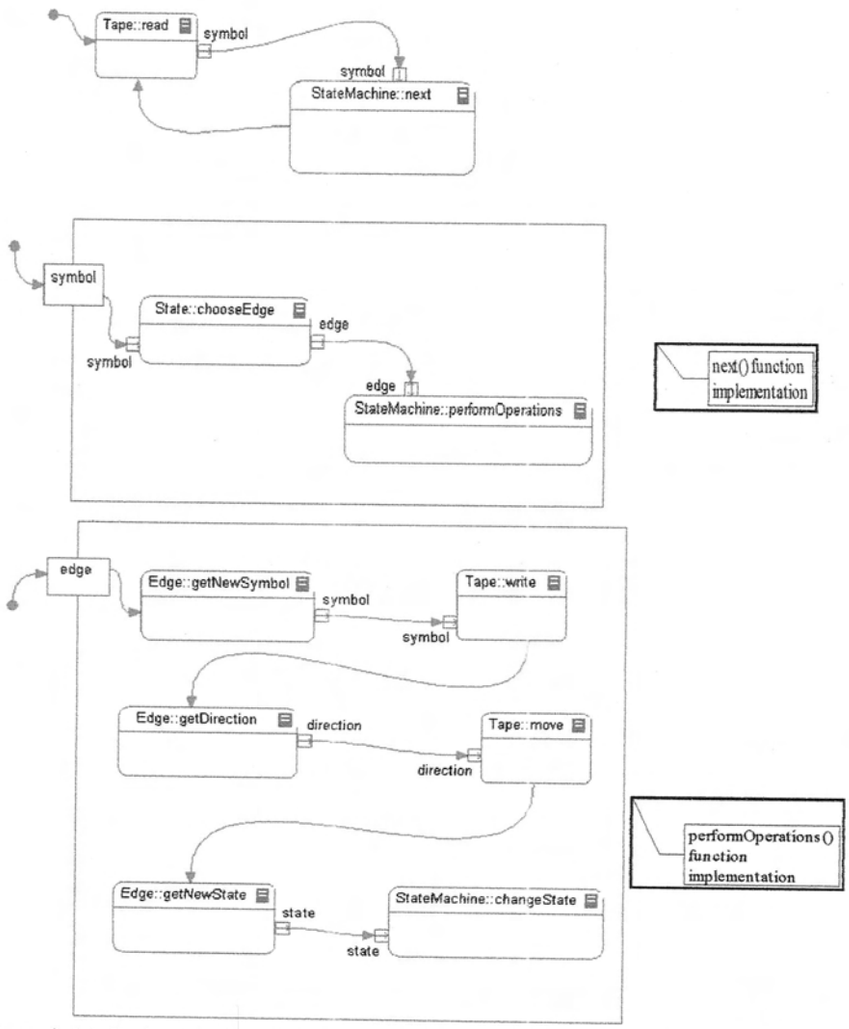

Figure 9.6.4: Turing Machine Behavioral Model in SysML Option 2 (Structured Activity Diagram)

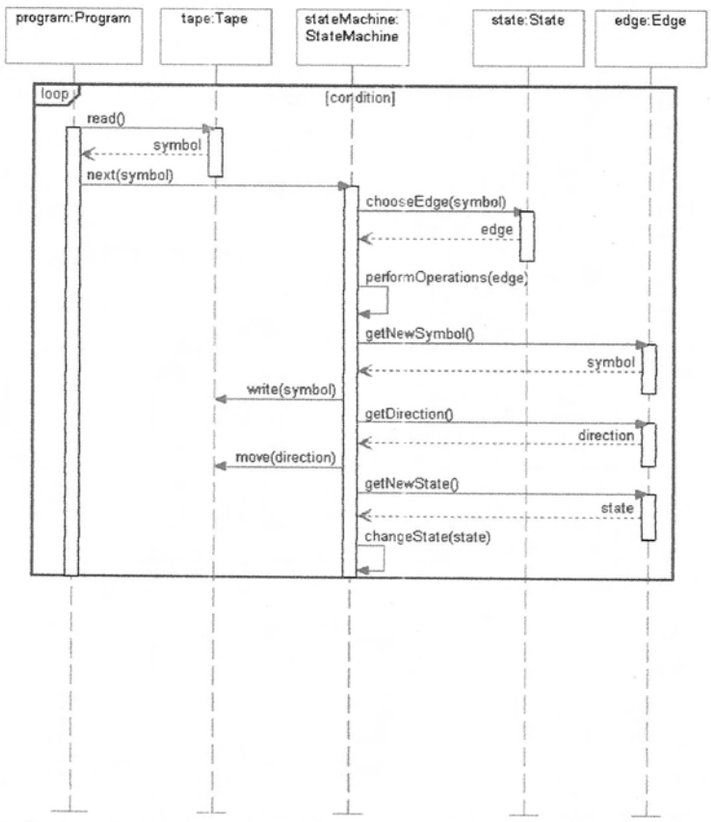

Figure 12: Turing Machine Behavioral Model in SysML Option 3 (Sequence Diagram).