MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

SYSML PARA ENGENHARIA SIMULTÂNEA DE SISTEMAS ESPACIAIS

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

Maiara Guimarães Flausino (UFSCar, Bolsista PIBIC/CNPq)
E-mail: maiara.flausino@lit.inpe.br

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

São José dos Campos – SP
Julho de 2013

Põe quanto és
No mínimo que fazes.

Fernando Pessoa

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

AGRADECIMENTOS

Ao Criador de tudo e à sua Mãe.

Aos meus pais, Luiz e Margareth, por tudo que me ensinaram e pelos muitos momentos de dificuldades que enfrentamos, mas que não impediram de sempre ter esperança em dias melhores.

As minhas irmãs, Luara e Moara, pelo incentivo de alcançar os sonhos.

Ao Professor Dr. Geilson Loureiro pela oportunidade de desenvolvimento deste trabalho, com o qual aprendi sobre a dinâmica do mundo acadêmico.

Ao CNPq pela bolsa de Iniciação Científica, a qual me permitiu acesso ao conhecimento.

Ao INPE pelo Programa Institucional oferecido, o qual me apoiou no desenvolvimento deste trabalho.

Ao LIT pela estrutura oferecida para o desenvolvimento deste trabalho.

RESUMO

Este trabalho tem por objetivo implementar o método de Engenharia Simultânea de Sistemas, usando a linguagem de modelagem de sistemas, *Systems Modeling Language* (SysML). Esse método de engenharia simultânea de sistemas foi inicialmente desenvolvido por Loureiro e evoluído desde 1999, o qual consiste no desenvolvimento simultâneo de produto e das organizações que implementam o ciclo de vida do produto ao longo dos processos de Engenharia Simultânea de Sistemas, quais sejam: análise de *stakeholders*, análise de requisitos, análise funcional e arquitetura de sistemas. O método deve ser aplicado em todos os níveis de abstração de um produto complexo, como um satélite, o qual foi originalmente desenvolvido usando técnicas de modelagem em Análise Estruturada e em *Unified Modeling Language* (UML). A partir de 2007, ferramentas computacionais que implementam a linguagem SysML passaram a estar disponíveis. A SysML é uma linguagem de modelagem gráfica descritiva de sistemas, construída a partir de estereótipos, os quais permitem customizar a modelagem. Um dos pilares da Engenharia Simultânea de Sistemas é a modelagem, especialmente a modelagem gráfica, que permite ao profissional ter uma visão do todo, bem como identificar cada relacionamento existente num dado sistema. Este trabalho aplicou o método, usando a linguagem SysML, no desenvolvimento de um picossatélite chamado CANSAT. Como conclusão tem se que a linguagem de SysML ainda precisa de adaptações. No entanto, ela já possui construções suficientes para implementação do método de forma a organizar os fluxos lógicos nele contido.

Palavras-chave: SysML, Engenharia Simultânea de Sistemas, Modelagem Gráfica, Sistemas Complexos.

1. INTRODUÇÃO

Este trabalho, desenvolvido no Laboratório de Integração e Testes (LIT), do Instituto Nacional de Pesquisas Espaciais (INPE), refere-se à compreensão do método de Engenharia Simultânea de Sistemas, desenvolvido por Loureiro (1999), e ao conhecimento da linguagem de modelagem descritiva de sistemas, *Systems Modeling Language* (SysML).

A Engenharia de Sistemas requer a visão do todo, de todas as fases do ciclo de vida do produto. Evidência disto é a evolução dos conceitos de Engenharia de Sistemas adotados pelas indústrias automobilística e aeroespacial no sentido de obter uma solução balanceada que considera variáveis, como o tempo de desenvolvimento, custo dos processos do ciclo de vida, gerenciamento de risco, desempenho do produto e, ainda, atende aos requisitos (LOUREIRO, 1999).

A evolução citada anteriormente consiste em analisar os *stakeholders*, os requisitos, o conceito funcional e a arquitetura de implementação, simultaneamente, para o produto, seus processos de ciclo de vida e suas organizações atuando em todas as camadas da estrutura de divisão de sistemas (LOUREIRO, 2010).

Um dos pilares da Engenharia Simultânea de Sistemas é a modelagem, especialmente a modelagem gráfica, a qual permite ao profissional ter uma visão sistêmica, bem como identificar cada relacionamento existente num dado sistema, ou seja, a modelagem permite que tanto os detalhes sejam conhecidos como também o todo.

O método consiste no desenvolvimento simultâneo de produto e de organizações que implementam as fases do ciclo de vida do produto ao longo dos processos de Engenharia Simultânea de Sistemas, quais sejam: análise de *stakeholders*, análise de requisitos, análise funcional e da arquitetura de sistemas. O método deve ser aplicado em todos os níveis de abstração de um produto complexo, como um satélite.

2. OBJETIVOS

Diante do exposto, este trabalho tem como objetivo principal facilitar o entendimento do método proposto por Loureiro (1999), desenvolvendo um *template* com base na linguagem SysML (*Systems Modeling Language*), através da realização de modelos experimentais, utilizando o *software IBM Rational Raphsody* 7.6.

Esta pesquisa tem também como objetivo consolidar os conhecimentos adquiridos durante o processo de estudo e pesquisa deste projeto, tanto da linguagem SysML como da abordagem da Engenharia Simultânea de Sistemas.

Para atingir o objetivo principal proposto, foram estabelecidos os seguintes objetivos específicos.

a) Realizar leituras críticas de referências bibliográficas sobre os temas em estudo (Engenharia Simultânea de Sistemas e linguagem SysML).
b) Realizar leituras críticas de notas de aulas das disciplinas Engenharia de Sistemas (LOUREIRO, 2011) e Modelagem de Sistemas (LOUREIRO, 2012).
c) Aprender a utilizar o *software IBM Rational Raphsody* 7.6.
d) Estudar as interfaces entre os softwares que possuem integração com o *IBM Rational Rhapsody* 7.6.
e) Pesquisar e analisar documentos, artigos científicos e tutoriais tanto sobre a linguagem SysML como sobre o *software IBM Rational Raphsody* 7.6.
f) Conhecer o Sistema CANSAT, para que seja possível a modelagem do template, com base em produto espacial.
g) Unificar os dados colhidos, analisando os resultados encontrados.
h) Modelar o *template*, usando todo o conhecimento adquirido durante o estudo e a pesquisa.
i) Preparar o relatório final do trabalho, propondo o *template* do sistema CANSAT.

3. MOTIVAÇÃO

A motivação para o desenvolvimento deste trabalho foi o aumento da necessidade do uso da Engenharia Simultânea de Sistemas na construção de produtos de vários setores da sociedade. Essa abordagem tem sido cada vez mais requisitada devido à evolução das tecnologias e ao crescimento da complexidade dos próprios produtos, como também ao aumento de riscos e de tempo gasto nas etapas de modelagem de um projeto.

O método de Engenharia Simultânea de Sistemas, proposto por Loureiro (1999), é aquele utilizado na pós-graduação do Instituto Nacional de Pesquisas Espaciais (INPE) para ensinar Engenharia de Sistemas. Esse método propõe a modelagem simultânea do produto e da organização ao longo dos processos de análise de *stakeholders*, de análise de requisitos, de análise funcional e de análise de implementação. A organização se refere às instituições que implementam os processos do ciclo de vida do produto (LOUREIRO, 2010).

Atualmente, os projetos são mais complexos, o que dificulta a visão sistêmica do produto. Para atender a essa nova demanda, vem sendo desenvolvida a linguagem SysML, a qual tem se tornado o padrão de modelagem de sistemas, tanto no mundo empresarial como no acadêmico e científico. Essa linguagem tem sido evoluída em conjunto com o OMG (*Object Management Group*) e o INCOSE (*International Council on Systems Engineering*).

No caso do Laboratório de Integração e Testes (LIT), o qual se enquadra neste contexto, existe um projeto aprovado, já em andamento, junto ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Agência Espacial Brasileira (AEB) para a implantação do Laboratório de Engenharia Simultânea de Sistemas (LSIS), que visa suprir essas necessidades encontradas cada vez mais no Laboratório. Esse laboratório deverá usar diversas ferramentas computacionais de Engenharia Simultânea de Sistemas tais como *Cradle* (ferramenta de Engenharia de Sistemas), *IBM Rational Rhapsody* (ferramenta de modelagem na linguagem SysML), *IBM Rational DOORS* (ferramenta de Engenharia e Gestão de Requisitos) e *MATLab/Simulink* (ferramenta de modelagem matemática).

4. METODOLOGIA

A metodologia utilizada neste estudo envolveu diferentes momentos. No início, com base na análise crítica da literatura científica pesquisada, foi feita a extração de conceitos do referencial teórico sobre a *Systems Modeling Language* (SysML) e do método de Engenharia Simultânea de Sistemas, proposto por Loureiro (1999). Estes ofereceram os fundamentos a partir dos quais o Sistema CANSAT foi modelado.

A realização deste trabalho ocorreu de agosto de 2011 a julho de 2013. Durante esse período de pesquisas e estudo continuo, através da revisão bibliográfica, leitura de artigos científicos da área, sobre a linguagem SysML, foram investigadas as suas características, os seus diagramas, as suas utilidades e as suas perspectivas para a descrição de sistemas complexos.

Para colocar em prática o aprendizado da linguagem SysML, foi necessário também o treinamento da ferramenta computacional *IBM Rational Rhapsody* 7.6, a qual se deu através de manuais de usuários, tutoriais virtuais e também da participação em fóruns *online*.

Durante esse processo, foi realizada uma vasta pesquisa sobre as ferramentas que modelam usando a linguagem SysML. O uso de todas elas ocorreu com a licença *trial* para testes quais sejam, *Enterprise Architecture (EA), IBM Rational Rhapsody, Magic Draw - SysML Plug-in, Modelio System Architecture, Papyrus UML for SysML e TopCased - SysML*, as duas últimas ferramentas são *open source*.

Como resultado apresentado neste trabalho de Iniciação Científica, o *software* utilizado foi o *IBM Rational Rhapsody* 7.6, por ser uma ferramenta com diversas funções, que ajudam na modelagem e na análise de um sistema, entre elas, rastreabilidade de requisitos, interface gráfica com acessibilidade e usabilidade, integração com o *IBM Rational DOORS*, geração de códigos e relatórios automaticamente e também por estar disponível no Laboratório de Engenharia Simultânea de Sistemas (LSIS), do Laboratório de Integração e Testes (LIT). Adicionalmente, foram realizadas discussões técnicas com o orientador sobre o assunto.

Foi preciso também realizar a identificação das interfaces da ferramenta computacional *IBM Rational Rhapsody* 7.6 com outros *softwares*, que também dão suporte ao método estudado neste trabalho, como o *IBM Rational DOORS*, o *Cradle* e o *MATLab/Simulink*.

O produto espacial escolhido para ser modelado foi o Sistema CANSAT, o qual foi utilizado como exemplo em situação de sala de aula. Este sistema é composto por uma estação terrena, um foguete e um picossatélite chamado CANSAT, artesanalmente construídos para a competição *7th Intercollegiate Rocket Engineering Competition (7th IREC).*

Esta pesquisa teve como resultado o *template* do método proposto por Loureiro (2010), usando a linguagem SysML com base no exemplo dado em aula, o Sistema CANSAT, o qual focaliza na modelagem do segmento da estação terrena desse Sistema. Muitas adaptações foram necessárias para a realização deste *template*, pois o método utilizado é baseado em Análise Estruturada, ou seja, os modelos refletem diretamente a linguagem estruturada, separando os registros (dados) das funcionalidades (procedimentos e funções).

5. PLANO DE ATIVIDADES

Atividade 1: MÉTODO: Obtenção e leitura da bibliografia específica e do material de treinamento relativo à Engenharia Simultânea de Sistemas Espaciais, desenvolvido por Loureiro (1999), disponível no site do curso CSE201-4 da pós-graduação do INPE.

Atividade 2: SYSML: Leitura das três referências básicas sobre a linguagem SysML.

a) HOLT, J.; PERRY, S. **SysML for systems engineering**. IET. Stevenage. 2008.
b) WEILKIENS, T. **Systems engineering with SysML/UML**: modeling, analysis, design. The MK/OMG Press. Elsevier. Amsterdam, 2006.
c) FRIEDENTHAL, S.; MOORE, A. STEINER, R. **A practical guide to SysML**: the systems modeling language. The MK/OMG Press. Elsevier. Amsterdam, 2009.

Atividade 3: RHAPSODY: Treinamento na ferramenta computacional *IBM Rational Rhapsody* 7.6.

Atividade 4: INTERFACES: Identificação de interfaces entre o *software IBM Rational Rhapsody* 7.6 e outras ferramentas, como *IBM Rational DOORS, Cradle* e *MATLab/Simulink.*

Atividade 5: TEMPLATE: Desenvolvimento do *template* de Engenharia Simultânea de Sistemas Espaciais, usando a linguagem SysML.

Atividade 6: EXEMPLO: Aplicação do *template* a um exemplo da área espacial, o Sistema CANSAT.

Atividade 7: RELATÓRIO FINAL 2012: Elaboração do relatório sobre as atividades da Iniciação Científica, referente ao primeiro período da bolsa PIBIC.

Atividade 8: SICINPE-2012: Seminário de Iniciação Científica e Iniciação em Desenvolvimento Tecnológico e Inovação. Apresentação desta pesquisa em pôster, em relação ao primeiro período da bolsa PIBIC.

Atividade 9: PRODUÇÃO CIENTÍFICA 2012: Escrita e apresentação de artigos e resumos científicos, referentes aos resultados obtidos no primeiro período da bolsa PIBIC.

a) Resumo científico aceito para o XVII Encontro de Iniciação Científica da UNITAU – ENIC/2012
b) Artigo Científico aceito para o XVI Encontro Latino-Americano de Iniciação Científica da UNIVAP – INIC/2012

Atividade 10: RHAPSODY: Aperfeiçoamento do uso do *software IBM Rational Rhapsody* 7.6.

Atividade 11: ATUALIZAÇÃO DO *TEMPLATE:* Correções da primeira versão do *template* como também a realização de estudos mais avançados em relação ao primeiro período da bolsa PIBIC.

Atividade 12: PRODUÇÃO CIENTÍFICA 2013: preparo e redação de artigos e resumos científicos, referentes aos resultados obtidos no primeiro e segundo períodos da bolsa PIBIC:

a) Resumo científico enviado para o XVIII Encontro de Iniciação Científica da UNITAU – ENIC/2013
b) Artigo Científico enviado para o XVII Encontro Latino-Americano de Iniciação Científica da UNIVAP – INIC/2013

Atividade 13: RELATÓRIO FINAL 2013: Elaboração do relatório sobre as atividades da Iniciação Científica referente ao primeiro e segundo períodos da bolsa PIBIC.

Atividade 14: SICINPE-2013: Seminário de Iniciação Científica e Iniciação em Desenvolvimento Tecnológico e Inovação. Apresentação desta pesquisa em pôster, em relação aos resultados alcançados no primeiro e segundo períodos da bolsa PIBIC.

6. RESUMO DAS ATIVIDADES REALIZADAS

Atividade 1:

Na primeira atividade foi fundamental o conhecimento e a leitura da bibliografia específica e do material de treinamento relativo à Engenharia Simultânea de Sistemas Espaciais, desenvolvido por Loureiro (1999), pois permitiu à estudante o entendimento do método. Esta atividade foi cumprida durante todo o período da bolsa, pois este está em constante evolução, e deverá sempre ser atualizado e divulgado para entendimento de todos os envolvidos em sua aplicação.

Atividade 2:

A leitura das três referências básicas sobre a linguagem SysML foi importante para aprender a modelar na linguagem SysML, mas o que trouxe mais praticidade foi a utilização dos *softwares open source*, bem como pesquisas realizadas, a participação no fórum *Google Group - SysMLForum* (sysmlforum@googlegroups.com), a leitura de revistas e artigos do INCOSE, do IEEE, e também leituras das especificações do OMG mais recentes. Todas essas atividades ajudaram a desenvolver o *template*.

Atividade 3:

O treinamento da ferramenta deu-se através da leitura de diversos manuais de usuários do próprio *software, IBM Rational Rhapsody* 7.6, e pesquisas realizadas na Internet. Muitos exemplos foram criados para conhecer melhor as funções dessa ferramenta computacional. A parte prática foi prejudicada, pois, o Laboratório de Engenharia Simultânea de Sistemas (LSIS) não possuía, até o mês de janeiro de 2012, essa ferramenta disponível para o desenvolvimento deste trabalho. Logo, seu desenvolvimento foi feito em parte com licença *trial* de 30 dias do *IBM Rational Rhapsody* 7.6. Por esse motivo, foram pesquisados outros *softwares,* os quais também modelam em linguagem SysML e possuem licença *trial*, quais sejam *Enterprise Architecture (EA), Modelio System Architecture, Papyrus UML for SysML e TopCased-SysML* (as duas últimas ferramentas são *open source*). Essas pesquisas serviram para realizar comparações entre os *softwares* apresentados anteriormente.

Atividade 4:

A identificação das interfaces entre as ferramentas computacionais *IBM Rational DOORS, Cradle, MATLab/Simulink* e o software *IBM Rational Rhapsody* 7.6 foram realizadas através de pesquisas e consultas aos manuais de usuário de cada ferramenta. A parte prática foi prejudicada, pois, o Laboratório de Engenharia Simultânea de Sistemas (LSIS) não possuía, até o mês de janeiro de 2012, essas ferramentas disponíveis para a integração desses *softwares*. Dessas ferramentas de integração, a única que oferece licença *trial* é a *Cradle*; por isso, não foi possível analisar a parte prática do projeto. Desta forma, só houve na prática a experiência de integração entre as ferramentas da IBM, pois são as únicas disponíveis no LSIS.

Atividade 5.

Muitas adaptações foram necessárias para a realização do *template*, usando a linguagem SysML, pois o método desenvolvido por Loureiro (1999) é modelado em Análise Estruturada, ou seja, os modelos refletem diretamente a linguagem estruturada, separando os registros (dados) das funcionalidades (procedimentos e funções). Com os dados obtidos pode-se afirmar preliminarmente que a linguagem SysML permite ao profissional que sejam criados novos tipos de diagramas, pois trata-se de uma linguagem não prescritiva, ao contrário da Análise Estruturada que é prescritiva. É importante lembrar que a linguagem SysML deve ser aprimorada, para garantir uma modelagem mais próxima da realidade dos sistemas complexos. Por isso, é preciso estar atento às especificações lançadas pelo OMG, desenvolvedor principal da linguagem, atualmente.

Atividade 6:

O produto espacial escolhido para ser modelado foi o Sistema CANSAT, o qual foi utilizado como exemplo em situação de sala de aula. Este sistema é composto por uma estação terrena, um foguete e um picossatélite, artesanalmente construídos para uma competição nos Estados Unidos "*7th Intercollegiate Rocket Engineering Competition (7th IREC)*".

Atividade 7:

Foi realizado e entregue o Relatório Final dentro do prazo para o PIBIC/INPE em julho de 2012.

Atividade 8:

Foi realizado e apresentado o pôster no dia proposto, 01 de agosto de 2012, no Atrium do Auditório Fernando de Mendonça – LIT/INPE.

Atividade 9:

Esta atividade foi incorporada ao plano de trabalho por se julgar importante dar maior visibilidade aos resultados encontrados. Sendo assim, foram preparados, escritos e apresentados um artigo e um resumo para encontros de Iniciação Científica em 2012.

Atividade 10:

Para o desenvolvimento completo desta pesquisa, houve a necessidade de aperfeiçoamento no uso do *software IBM Rational Rhapsody* 7.6, ou seja, conhecer melhor suas funções para a estruturação adequada do *template* em desenvolvimento. Assim, foram modelados novos exemplos, usando a abordagem da Engenharia Simultânea de Sistemas.

Atividade 11:

Nesta atividade buscou-se realizar as correções no *template*, como também avançar no desenvolvimento do trabalho objeto desta bolsa.

Esses estudos incluíram criação de novos estereótipos de relações para modelar o produto espacial proposto.

Atividade 12:

Esta atividade também foi incorporada ao plano de trabalho por se julgar importante dar maior visibilidade aos resultados encontrados, sendo assim, foi preparado e escrito um artigo e um resumo para encontros de Iniciação Científica em 2013.

Atividade 13:

Esta atividade trata-se da elaboração e revisão deste relatório final entregue no dentro do prazo para o PIBIC/INPE em julho de 2013.

Atividade 14:

Esta atividade foi realizada e será apresentada no dia proposto, 31 de julho de 2013.

7. CRONOGRAMA DE ATIVIDADES

O cronograma seguirá as mesmas datas propostas no início deste trabalho, uma vez que, a pesquisa caminha de forma satisfatória para a finalização deste projeto de Iniciação Científica dentro do prazo já previsto.

Atividade	2011					2012											2013						
	Ago.	Set.	Out.	Nov.	Dez.	Jan.	Fev.	Mar.	Abr.	Maio	Jun.	Jul.	Ago.	Set.	Out.	Nov.	Dez.	Jan.	Fev.	Mar.	Abr.	Maio	Jun.	Jul.
1																								
2																								
3																								
4																								
5																								
6																								
7																								
8																								
9																								
10																								
11																								
12																								
13																								
14																								

Tabela 1 - Cronograma de atividades.

8. FUNDAMENTACÃO TEÓRICA

Esta pesquisa estudou a linguagem SysML, uma vez que, essa segue a tendência da sua percussora, a linguagem UML. A linguagem SysML vem se tornando padrão para modelar sistemas complexos, que incluem *hardwares*, *softwares*, pessoas e troca de material, energia e informação.

Por isso é importante saber utilizar o paradigma da linguagem SysML para que se possa modelar tanto a estrutura como o comportamento de um sistema, para que os requisitos sejam atingidos ao final do projeto, pois muitas vezes a formulação de um problema é mais importante do que a sua solução (Albert Einstein).

Para conhecer e saber aplicar essa linguagem foi preciso investigar artigos e resumos científicos que abordassem o uso dessa modelagem, alem do treinamento da ferramenta *IBM Rational Rhapsody* 7.6, através de pesquisas e tutorias disponíveis na Internet.

Figura 1- Mapa conceitual da linguagem SysML.

Fonte: Flausino, 2013.

8.1. Sistemas

De acordo com Halligan (2013), um sistema é um conjunto de elementos, os quais são organizados a fim de operarem integradamente, para atingir os requisitos estabelecidos.

Quanto maior for o número de relações, mais complexo será o sistema, uma vez que, a sua complexidade depende da conexão entre cada um de seus elementos.

E cada elemento, por sua vez, também tem a sua complexidade, a qual depende dele mesmo e de sua relação com os outros elementos do sistema, pensando sistematicamente, cada elemento também pode ser considerado como um sistema ou também pode ser classificado como um subsistema.

Para ser entendido, um sistema deve ser analisado e estudado de forma holística, a qual visa compreender toda a sua extensão e também de maneira sistêmica, a qual tem como objetivo entender e identificar cada ligação do sistema, ou seja, as suas relações internas.

Os sistemas complexos, abordados por este trabalho, referem-se àqueles que resultam de um projeto de Engenharia Simultânea de Sistemas, que recaem em uma abordagem recursiva, na qual cada sistema pode ser analisado interativamente como resultante da interligação de subsistemas, cada um destes também estruturado na forma de sub-subsistema, e assim por diante, isto é, existe uma hierarquia a ser seguida no desenvolvimento de produtos, no qual há níveis de abstrações diferentes para a solução do projeto.

8.2. Engenharia de Sistemas

Engenharia de Sistemas é uma abordagem interdisciplinar e colaborativa para solucionar problemas complexos, que torna possível a concretização de sistemas de elevada complexidade (INCOSE, 2012).

Em outras palavras, trata-se de uma abordagem multidisciplinar, a qual integra diferentes especialidades do conhecimento, formando um processo de desenvolvimento estruturado que se estende da fase conceitual à fase operacional de um sistema. Portanto, considera tanto as questões de ordem econômica quanto as de ordem técnica, com o objetivo de gerar produtos de qualidade que atendam principalmente às necessidades dos *stakeholders*, diminuindo os impactos negativos durante o desenvolvimento.

O principal foco da Engenharia de Sistemas é definir as necessidades dos *stakeholders*, bem como as funcionalidades requeridas por eles. Essa abordagem tem como objetivo capturar as necessidades dos *stakeholders,* realizando a documentação sistemática dos requisitos, abordando a síntese de projeto e a etapa de validação de forma a considerar o problema por completo. Em seguida essas necessidades serão transformadas em requisitos, os quais serão validados posteriormente com os próprios *stakeholders* para garantir a satisfação deles em relação ao produto.

O termo Engenharia de Sistemas surgiu em um contexto computacional, mas atualmente ela engloba diversas outras áreas do conhecimento (HALL, 1962). Essa abordagem propõe solucionar problemas de interfaces entre os componentes de um sistema complexo, ou seja, visa combater não só a complexidade, como também a falta de entendimento entre os especialistas de cada subsistema.

A evolução dessa Engenharia, que ainda hoje continua, compreende o desenvolvimento e a identificação de novos métodos e técnicas de análise e desenvolvimento de produtos complexos. Esses novos métodos buscam melhorar a compreensão do sistema como um todo, já que, os processos de Engenharia de Sistemas tendem a serem documentados a partir de várias técnicas que muitas vezes são imprecisas e inconsistentes.

8.3. Engenharia Simultânea

A abordagem de Engenharia Simultânea antecipa para as etapas iniciais do desenvolvimento do produto os requisitos relativos ao ciclo de vida produto, bem com os relativos ao ciclo de vida das organizações que implementam esses produtos. Salvo que a antecipação de cada requisito ocorre separadamente para fase inicial do projeto, ou seja, a abordagem sistêmica não é necessariamente considerada neste processo.

A Engenharia Simultânea reconhece o impacto, do ciclo de vida do produto em um projeto sistêmico, em virtude disso, aborda cada fase do ciclo de vida como uma etapa isolada. De modo que, é tradicionalmente aplicada no desenvolvimento de elementos e não de sistemas. (PRASAD, 1996)

Com a prática da Engenharia Simultânea foi possível transformar o cenário espacial, o qual será descrito a seguir.

ANTES	**ATUALMENTE**
Desempenho a qualquer custo.	Desempenho, risco, custo e cronograma.
Aversão a riscos.	Gestão de risco.
Foco em documentação.	Foco em *trade-off*.
Fases rígidas.	Gestão e análise de *stakeholders*; mais simulação do que testes físicos.
Engenharia Sequencial.	Engenharia Simultânea.

Tabela 2 - Comparação do cenário espacial.

Fonte: Loureiro, 2011.

8.4. Método proposto por Loureiro

Loureiro (1999), definiu que Engenharia Simultânea de Sistemas é a antecipação dos requisitos para a fase inicial do projeto, de uma forma relacional, ou seja, a visão do todo é levada em consideração para que as antecipações não causem nenhum prejuízo ao produto, depois que os elementos sejam integrados.

Desta forma, a Engenharia Simultânea de Sistemas também é uma abordagem multidisciplinar, desenvolvida para solucionar problemas de integração das partes do sistema complexo, levando em consideração o todo.

O método apresentado por Loureiro (1999), Figura 2, possui três tipos de análises: a análise de requisitos, a análise funcional e a análise física. E possui três dimensões a serem integradas: o produto, o processo e a organização.

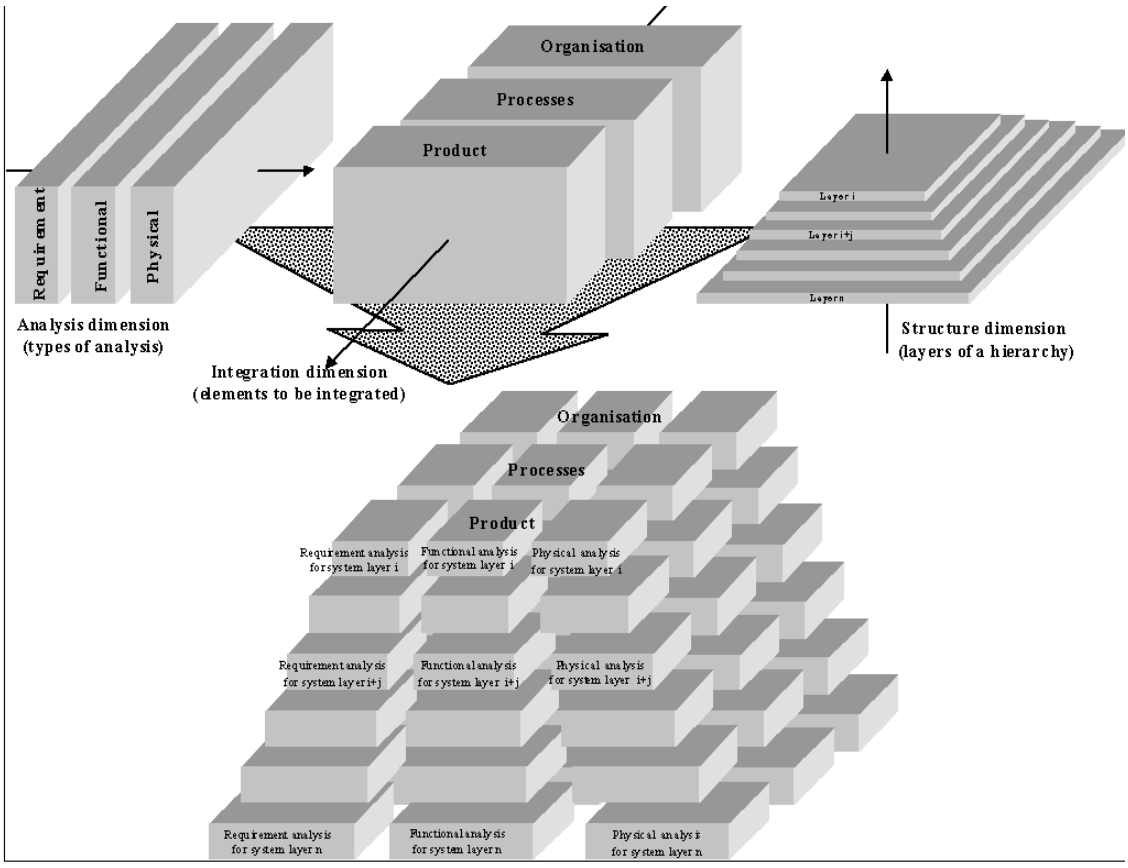


Figura 2 - Método de Loureiro (1999).

Fonte: Loureiro, 1999.

A Figura 3 apresenta o método de Loureiro (2010) evoluído, o qual realiza simultaneamente para produto e organização mais um tipo de análise, a análise de *stakeholders*, a qual tem como objetivo integrar o próprio *stakeholders* durante todo o processo, sendo assim a solução do sistema será baseada nos *stakeholders*.

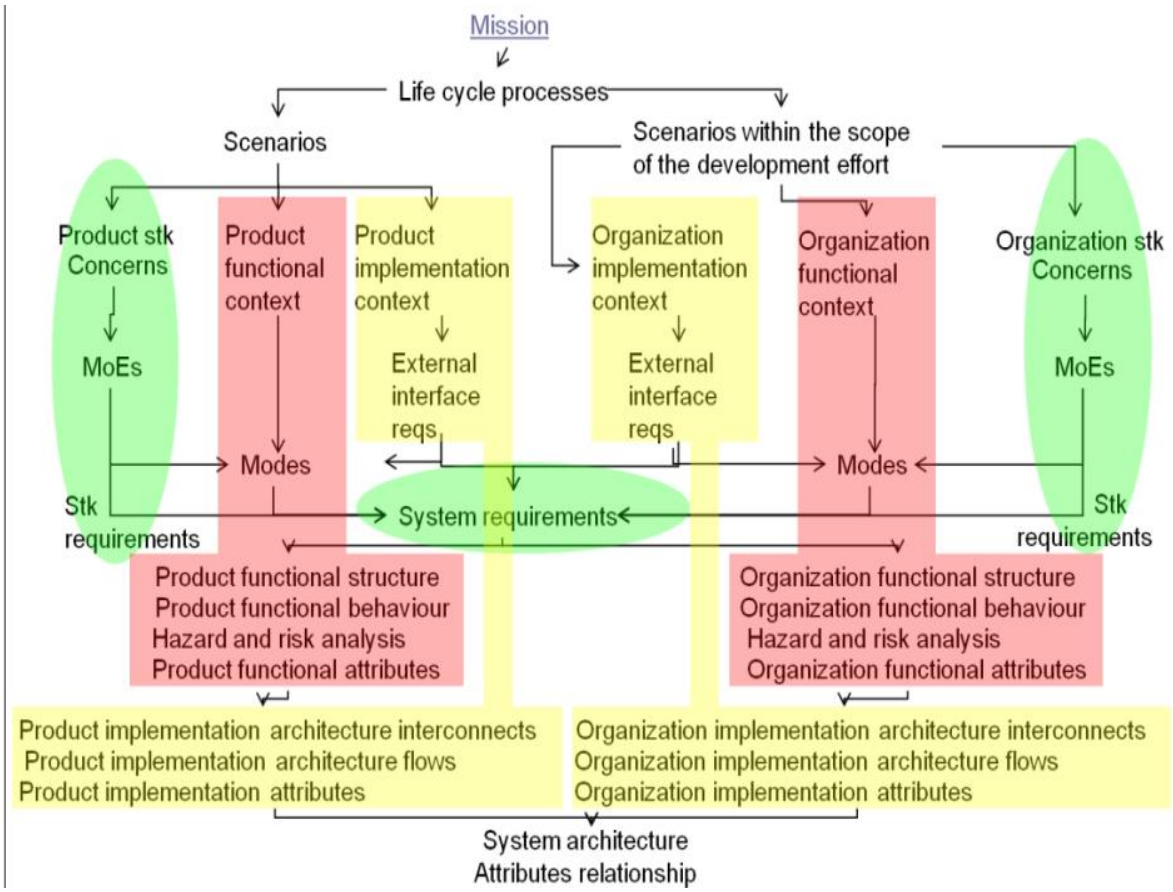


Figura 3 - Método de Loureiro (2010).

Fonte: Loureiro, 2010.

Com a evolução do método é possível perceber que a iteração existente na análise de requisitos auxilia a realimentação de requisitos que atendam as necessidades dos *stakeholders*. Assim, os requisitos são identificados, iterativamente, na interação entre a análise de requisitos e a análise funcional, sendo que a validação do sistema ocorre com a satisfação do *stakeholders*, ou seja, com a confirmação de que o produto atende os requisitos de *stakeholders*.

Portanto, o método passou a ter quatro tipos de análises: a análise de requisitos, a análise funcional, análise física e a análise de *stakeholders*.

8.5. Modelagem gráfica

As informações são veiculadas por diversas representações, e cada uma possui sua própria semântica e sintaxe, sendo que uma dessas representações é a modelagem gráfica, a qual facilita o processamento e a apresentação de informações dos sistemas complexos. Pois um modelo é a simplificação da realidade, o qual também pode ser entendido como uma abstração de um ambiente em estudo, auxiliando na visualização do todo.

O diagrama é uma representação gráfica de um conjunto de elementos, sendo que eles podem ser divididos em dois tipos quais são: a) Comportamento, que denota a dinâmica dos elementos, como a operação em um sistema. b) Estrutura, a qual denota a estática dos elementos.

O uso de diagramas é uma técnica aprovada e aceita pelos profissionais que os utilizam, tanto no mundo empresarial com acadêmico, que vai desde modelos matemáticos e computacionais, até os protótipos.

Os diagramas permitem que o sistema seja compreendido por todos envolvidos em um projeto. Pois a modelagem permite: a) Visualizar um sistema inteiramente. b) Especificar a estrutura e o comportamento de um sistema. c) Proporcionar um *template* para construção de um sistema (objetivo deste trabalho). d) Documentar as decisões tomadas em cada fase do projeto.

Os princípios de modelagem, segundo Paim (2009), são apresentados a seguir:

a) Aderência: aproximação do modelo ao funcionamento do processo ou do sistema.
b) Clareza: simplicidade e objetividade, que ajuda na compreensão do modelo.
c) Comparabilidade: consistência entre os níveis de abstração de um sistema.
d) Relevância: o modelo deve ser capaz de responder: o que deverá ser feito? Como deverá ser feito? Quem fará? Onde será executado?

É através da modelagem, que o fluxo de material, energia e informação é identificado. Já que, as formas gráficas são mais puras, por serem mais fieis ao raciocínio lógico (MEDINA, 2005).

Diversas vezes a modelagem gráfica gera desconforto porque é mais fácil e comum as pessoas expressarem-se por palavras, porém uma das grandes vantagens de utilizar os modelos gráficos é substituir um grande número de palavras por convenções de desenhos, isto é, símbolos (MEDINA, 2005).

Desta forma, os modelos não só podem, como devem ser feitos, a fim de visualizar e controlar o todo, gerenciando os riscos latentes em um projeto, uma vez que, quanto mais complexo for o sistema, mais importante se torna a sua modelagem.

Atualmente, com o auxílio da informática ficou mais fácil de corrigir possíveis erros na modelagem, sendo que a escolha da ferramenta computacional faz grande diferença na qualidade da modelagem do sistema ou do processo, pois é importante selecionar uma ferramenta que ajude o modelador a realizar simplificações consistentes, sempre reaproveitando o que for possível para ganhar tempo na execução da modelagem.

Por isso, é fundamental fazer uma pesquisa de *softwares* que atendem as reais necessidades do projeto, pois a abordagem usada na modelagem deve ser escolhida levando em consideração o sistema que será modelado, a interface do *software*, e a linguagem a ser aplicada, garantindo que a modelagem seja fiel a realidade do produto.

Para obter uma modelagem fiel ao um sistema complexo é preciso uma ferramenta que de suporte ao projeto. Pois, apenas com um *software* é possível realizar, por exemplo: a) Controle de versões de um modelo. b) Análises estruturais e comportamentais do sistema em estudo. c) Rastreamento de requisitos tanto de *stakeholders* como os de sistemas. d) Captura de novos requisitos, pois a partir dos modelos é possível extrair novos requisitos do sistema modelado.

8.5.1. As quatro bases da modelagem

Dentro do contexto de Engenharia Simultânea de Sistemas têm-se quatro pilares que sustentam a modelagem, Figura 4, sendo elas:

a**)** Ciclo de vida: O modelador deve estar ciente sobre qual será o modelo de ciclo de vida (V; Espiral; Cascata) a ser utilizado no projeto.

b) Abordagem: A abordagem da modelagem deve ser escolhida segundo o sistema que será modelado, para que o resultado seja fiel a realidade do produto em estudo. A rigidez sintática e semântica faz parte das linguagens de modelagem, porque garante ao modelo que não haja mais de uma interpretação possível, ou seja, não gere ambiguidade.

c) Linguagem: Deve-se pesquisar qual é a linguagem mais apropriada para modelar o sistema a ser desenvolvido.

d) *Software*: Deve-se ter domínio sobre a ferramenta computacional escolhida para modelar o produto conhecendo suas funções disponíveis, para garantir, que se tenha ao final, um trabalho de alta qualidade.

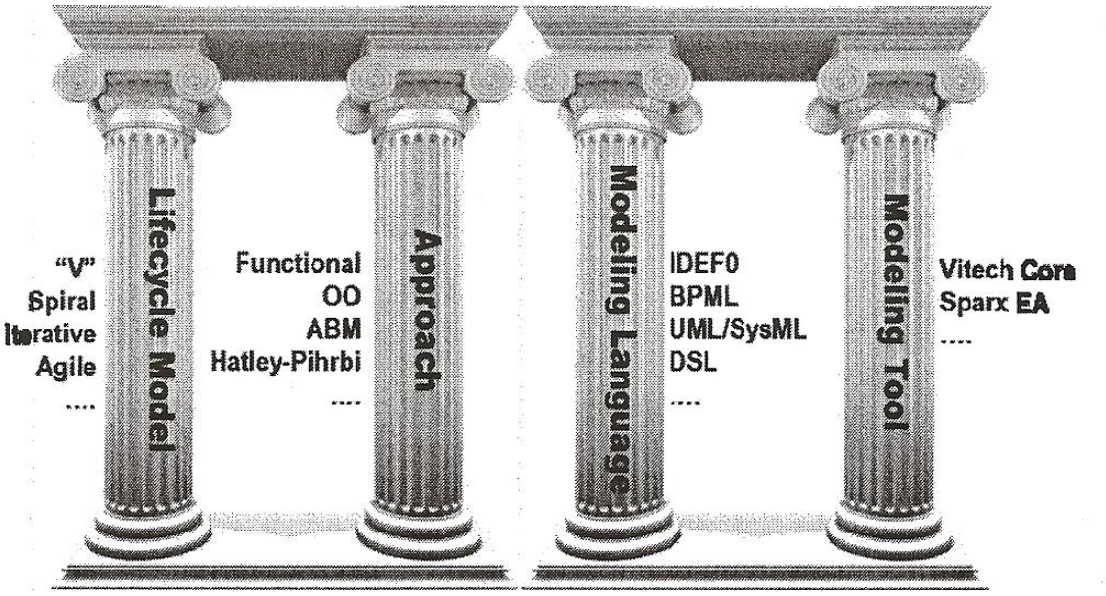


Figura 4 - As quatro bases da modelagem.

Fonte: INCOSE, 2011.

8.5.2. Análise Estruturada

No final da década de 1970, a Análise Estruturada, abordagem *bottom-up*, permitiu a especificação de requisitos lógicos de um sistema em um modelo gráfico de alto nível de abstração, possibilitando que usuários compreendessem o que estava documentado nos diagramas, além de garantir que a arquitetura do sistema fosse mapeada.

Esse tipo de modelagem refere-se à construção de diagramas com fluxo de dados e informações sendo dividida em elementos funcionais, os quais descrevem o que deve ser construído, isto é, os diagramas podem representar: os dados utilizados por um sistema; os fluxos que transportam; os processos que os transformam.

A Análise Estruturada contém modelos que possibilita a demonstração de fluxo de informação, usando símbolos, juntamente com um dicionário de dados como complemento aos modelos de fluxo de informação. A técnica estruturada possui também a modelagem comportamental, a qual representa a dinâmica de um sistema, descrevendo seus estados e eventos.

Um diagrama muito conhecido e utilizado para expressar a troca de dados é o Diagrama de Fluxo de Dados (DFD), o qual descreve o fluxo de informação e transformações que são realizadas no sistema conforme os dados se movem da sua entrada para a sua saída. Com o diagrama DFD pode representar todos os níveis de abstração.

Para auxiliar a modelagem em Análise Estruturada, como, rastrear e gerenciar mudanças no sistema, as ferramentas computacionais tornaram-se fundamentais para garantir a qualidade da modelagem. Pois estas constroem uma hierarquia interna de forma que cada bolha "pai" e seus "filhos" sejam automaticamente associados entre si.

Há ainda na Análise Estruturada extensões para modelar sistemas em tempo real, os quais são orientados pelo controle dos eventos com interação ao mundo real em um determinado período. Uma dessas extensões é a do Hatley & Pirbhai, a qual visa a representação e a especificação dos aspectos orientados ao controle do *software*, criando um modelo de sistema de tempo real.

8.5.3. *Unified Modeling Language* (UML)

A *Unified Modeling Language* (UML) é uma linguagem de modelagem gráfica estabelecida no campo de desenvolvimento de *software*, sendo um padrão internacional especificado pelo *Object Management Group* (OMG) e também aceita como padrão ISO/IEC 19501 (WEILKIENS, 2007).

Como a linguagem UML foi projetada para *software*, é ineficaz para aplicação em outros sistemas, como, sistemas espaciais, mecânicos, médicos e financeiros. Outro problema encontrado é o fato da linguagem UML ter sido desenvolvida para análise e validação de *softwares* ao passo que a modelagem dos outros sistemas envolve outras disciplinas além da Engenharia de S*oftware*, ou seja, são interdisciplinares.

Várias iniciativas de padronizar os processos de Engenharia Simultânea de Sistemas sugiram, mas até o início dos anos 2000, nenhuma linguagem de modelagem havia sido criada, exigindo a utilização de diversas linguagens em um mesmo projeto, o que dificultava a comunicação entre os profissionais envolvidos.

Devido a essa necessidade de modelar outros sistemas, em 2001, o OMG juntamente com o *International Council on Systems Engineering* (INCOSE) decidiu criar um novo padrão da linguagem para sistemas complexos, foi quando surgiu a extensão da UML, a *Systems Modeling Language* (SysML), específica para modelar os sistemas complexos.

A linguagem UML foi escolhida, pois atendia a alguns requisitos, já era amplamente utilizada e aceita pelos profissionais da área e também possuía muitas ferramentas que davam suporte a ela, o que facilitaria as adaptações para essa nova necessidade. Alem de sua propriedade de extensão, pois através do mecanismo de estereótipos é possível criar novas definições e adaptações para um domínio mais específico.

A extensão da linguagem UML derivou a linguagem SysML, sendo que foi preciso a criação de dois novos tipos de diagramas: paramétrico e de requisito, bem como houve a necessidade da omissão de alguns elementos da linguagem UML específicos para o desenvolvimento de *software*.

8.5.4. *Systems Modeling Language* (SysML)

A *Systems Modeling Language* (SysML) é uma linguagem de modelagem gráfica de propósito geral, para a análise, especificação, projeto, verificação e validação de sistemas complexos, sendo que nesses sistemas podem estar incluídos *hardwares*, *softwares*, dados, pessoas, procedimentos, facilidades e outros elementos de sistemas naturais (FRIEDENTHAL et al, 2009).

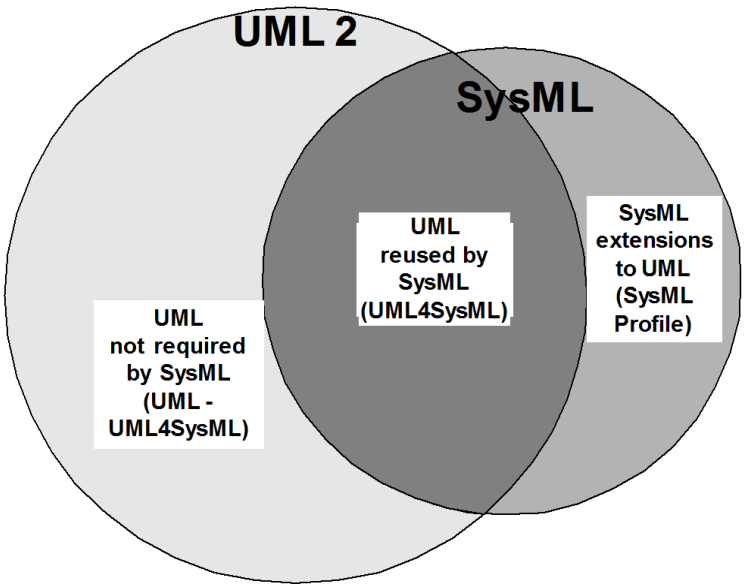


Figura 5 - Relação entre as linguagens UML e SysML.
Fonte: Friedenthal et al, 2009.

A linguagem SysML pode ser usada para modelar produtos e processos, elementos presentes no método desenvolvido por Loureiro (1999).

Essa linguagem foi proposta para padronizar a modelagem do projeto de um sistema complexo, ela ainda está em fase de disseminação e aceitação para a comunidade científica e empresarial, pois é uma linguagem gráfica nova e encontra-se em fase de desenvolvimento pelo OMG em parceria com o INCOSE desde 2001, quando criaram um grupo de trabalho que definiu os requisitos de uma linguagem de modelagem para Engenharia de Sistemas.

Esses requisitos foram publicados em 2003 em uma chamada para propostas (UML for *Systems Engineering Request for Proposal* - UML for SE RFP). Em seguida foi organizado o *SysML Partners*, um grupo de trabalho composto por representantes do setor industrial e por produtores de ferramentas computacionais.

Esse grupo iniciou a elaboração da linguagem *SysML Open Source*, uma linguagem que respondesse aos requisitos especificados na UML for SE RFP. A versão "zero" da linguagem SysML foi publicada em 2004, enquanto que a primeira versão (SysML 1.0), foi proposta no final de 2005. A versão pública formal (SysML 1.2) foi publicada pelo OMG em junho de 2010. Atualmente, encontra-se na versão SysML 1.3, a qual foi publicada em junho de 2012.

Seguem os participantes do *SysML Partners*:

Industry and government
American Systems
BAE Systems
The Boeing Company
Deere & Company
EADS Astrium GmbH
Eurostep Group AB
Lockheed Martin Corporation
Motorola, Inc.
National Institute of Standards and Technology (NIST)
Northrop Grumman Corporation
oose.de Dienstleistungen für innovative Informatik GmbH
Raytheon Company
Thales
Tool vendors
ARTiSAN Software Tools
EmbeddedPlus Engineering
Gentleware AG
IBM
I-Logix, Inc.
Mentor Graphics
PivotPoint Technology Corporation
Sparx Systems
Telelogic AB
Vitech Corp
Academia
Georgia Institute of Technology
Liaison organizations
INCOSE
ISO 10303-233 (AP233) Working Group

Tabela 3 - *SysML Partners.*

Fonte: Holt & Perry, 2011.

A linguagem SysML consegue especificar, analisar e verificar os sistemas complexos devido a sua sintaxe e seus símbolos, garantindo que o modelo seja consistente nas informações que representam.

A Tabela 4 mostra que a linguagem SysML é capaz de suprir as deficiências encontrados nos três males da Engenharia de Sistemas.

Os três males da ES	**Os efeitos dos três males da ES**	**Aplicação da linguagem SysML**
Complexidade	Variedade	Análise
Falta de compreensão	Desordem	Estrutura/comportamento
Falta de comunicação	Peças soltas	Integração

Tabela 4 - Aplicação da linguagem SysML.

Fonte: Flausino, 2013.

A complexidade dos sistemas gera diferentes interpretações sobre o mesmo produto e a linguagem SysML através de seus diagramas garante que as análises possam ser feitas, possibilitando a mesma interpretação por todos os envolvidos.

A falta de comunicação gera desordem na condução do projeto, como, problemas ao integrar os subsistemas do produto. Para resolve essa situação, a linguagem SysML possui digramas que permitem que o comportamento e a estrutura do sistema sejam avaliados através dos modelos.

Diante do apresentado, pode-se dizer que, a principal motivação para usar a linguagem SysML é conseguir reunir e integrar todos os requisitos, isto é, é preciso uma modelagem sólida que aceite modificações e atualizações.

Por isso, essa linguagem surgiu com o propósito de eliminar as interpretações equivocas e ambíguas aos sistemas complexos, pois possui elementos necessários para satisfazer as necessidades da Engenharia de Sistemas.

Para que esse objetivo seja cumprido, a aplicação da linguagem SysML deve ser feita segundo as suas próprias regras, para que realmente tenha-se os diagrama dessa linguagem, observe a Tabela 5, a seguir.

SysML + consistência = MODELO	SysML – consistência = desenho

Tabela 5 - Linguagem SysML

Fonte: Holt; Perry, 2011.

Isso se deve ao fato que a linguagem SysML derivou-se da linguagem UML, a qual também vem sendo aperfeiçoada, atualmente encontra-se na versão 2.4.1, esta por sua vez, é baseada no paradigma de orientação a objeto, o qual aumenta a velocidade no desenvolvimento dos programas devido a reutilização dos objetos, sendo assim a linguagem SysML também reaproveita os diagramas, fator que ajuda a modelagem a ficar concisa e menos suscetível a erros, possibilitando mais análises e simulação a testes físicos, gerando grande economia e mais segurança à organização que dão suporte ao ciclo de vida do produto.

Na Tabela 6 é possível observar a comparação entre as linguagens UML e SysML.

UML diagrams	SysML diagrams
Class diagram	Borrowed and adapted with the concept of blocks within the Block Definition diagram
Object diagram	N/A
Package diagram	Borrowed
Component diagram	N/A
Composite Structure diagram	Borrowed and adapted with the concept of blocks within the Internal Block diagram
Deployment diagram	N/A
Use case diagram	Borrowed as-is
Activity diagram	Borrowed and extended
Interaction diagram	N/A
Communication diagram	N/A
Interaction Overview diagram	N/A
Sequence diagram	Borrowed and adapted
Timing diagram	N/A
State machine diagram	Borrowed and adapted
N/A	Requirement diagram
N/A	Parametric diagram

Tabela 6 - Comparação entre as linguagens UML e SysML.

Fonte: INCOSE, 2011.

O fato que aponta a popularização da linguagem SysML é o estudo, que busca a sua integração com a Modelica, ou seja mais um incentivo para aprender a linguagem SysML, uma vez que, a Modelica já é amplamente utilizada na indústria, principalmente automotiva.

O artigo publicado pelo INCOSE (2012), de Pihera & Ender (2012), *Applying systems Engineering to improve extracorporeal membrane oxygenation therapy*, é importante de ser citado, pois demonstra a importância que a linguagem SysML desempenha ao integrar varias área do conhecimento, ou seja, ela é capaz de supri a lacuna de comunicação que existe entre as áreas diversas, no artigo, por exemplo, as áreas trabalhadas são a Medicina e a Engenharia.

Essa pesquisa demonstra os benefícios para ambas as partes, proporcionando melhores condições para o desenvolvimento do trabalho em conjunto. Por se tratar de uma linguagem adaptável mesmo os leigos são capazes de compreender os sistemas no nível mais alto de abstração.

As principais funcionalidades e vantagens da linguagem SysML são: a) Gerenciamento de complexidade através da modelagem de pacotes. b) Comunicação entre os *stakeholders* e profissionais da organização que desenvolverá o produto. c) Redução de ambiguidade e erros nos modelos. d) Rastreabilidade de requisitos para validação futura. e) Possibilidade de simulações, uma vez que, modelos podem ser simulados, documentos não.

A linguagem SysML providencia ricos modelos que dão suporte a Engenharia Simultânea de Sistemas. Essa linguagem não tem por objetivo ser a melhor e única linguagem para resolver todos os problemas de modelagens de um sistema, isto é, ela não foi desenvolvida para simular os modelos, isso cabe ao *software* no qual será modelada. A simulação está relacionada à ferramenta computacional e não a linguagem em si. A linguagem SysML simplesmente captura os dados, possibilitando a simulações desses (LANE & BOHN, 2012).

Na Figura 6 é possível observar como estão organizados os noves tipos de diagrama da linguagem SysML.

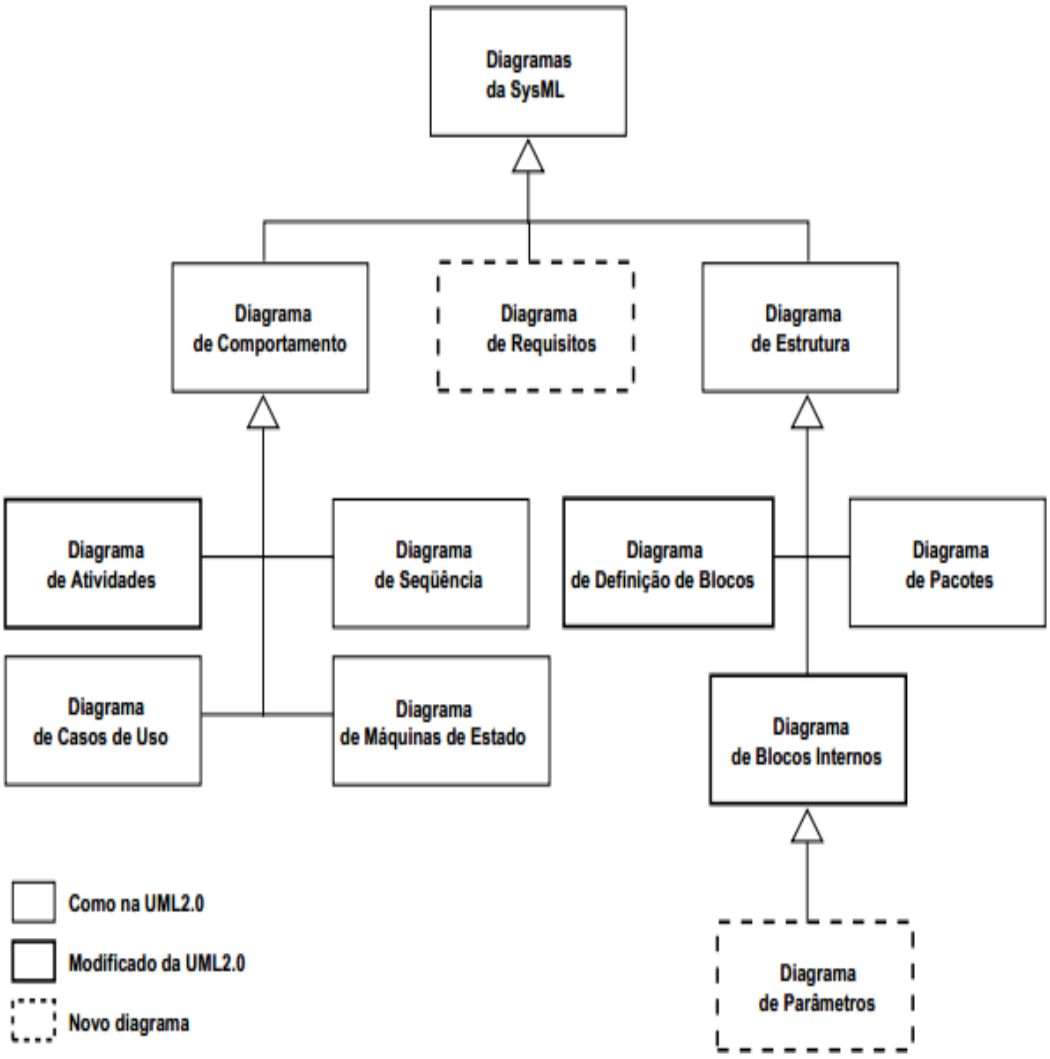


Figura 6 - Diagramas da linguagem SysML.

Fonte: Flausino, 2012.

A seguir, serão apresentados todos os diagramas da linguagem SysML e suas respectivas características.

8.5.4.1. Diagrama de atividade

O diagrama de atividade mostra o comportamento do sistema, incluindo controles e fluxos de dados, o qual pode ser utilizado para fazer a análise funcional comumente usado pelos engenheiros de sistemas.

Como esse diagrama modela o comportamento sistema, isto é, os aspectos dinâmicos do sistema, ele representa a concorrência e as ramificações de atividades, que podem ser os caminhos alternativos, tomados com base em alguma expressão boleana de forma linear ou sequencial. Seu principal objetivo é exibir as etapas de um processo, com enfoque no fluxo de material, energia e informação de uma etapa para outra.

Existem dois modos de criar o diagrama de atividades:

a) Fluxo de trabalho: especifica, constrói e documenta processos.

b) Operação: empregado como fluxograma, pois contem ramificações, bifurcações e estados de união. Representa os fluxos conduzidos por processamentos. É essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra. Comumente isso envolve a modelagem das etapas seqüenciais em um processo.

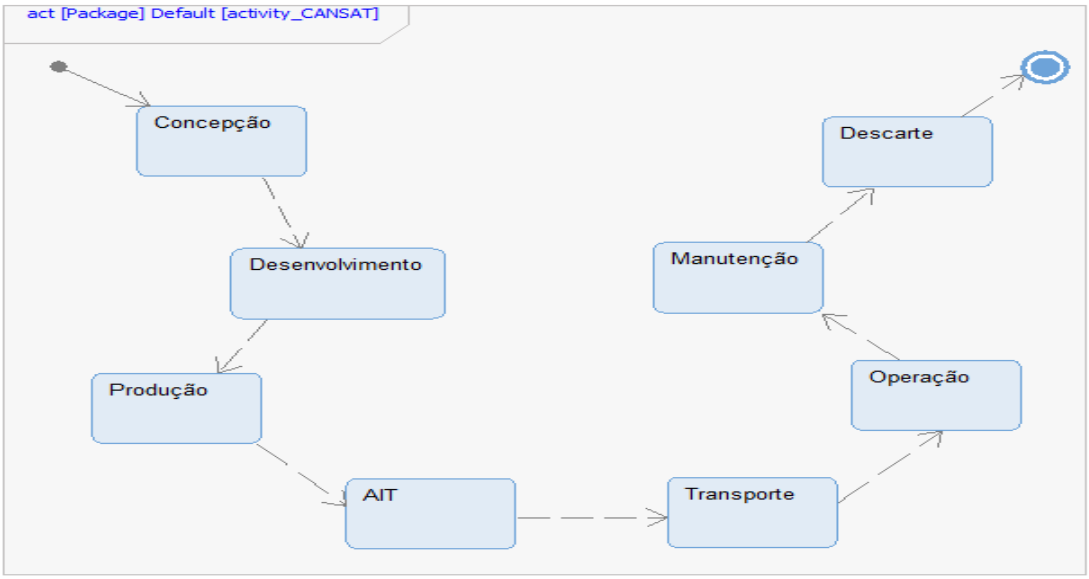


Figura 7 - Diagrama de atividade da linguagem SysML.

Fonte: Flausino, 2013.

8.5.4.2. Diagrama de sequência

O diagrama de sequência apresenta o comportamento de um sistema através das interações entre os seus elementos, enfatizando a ordenação temporal de suas atividades, através das mensagens que são trocadas entre as atividades do sistema. Entende-se por mensagens os serviços solicitados de uma atividade a outra, e as respostas desenvolvidas para as solicitações.

Esse diagrama é usado para análise e *design* de produtos complexos, pois descreve de forma simples e lógica como as atividades se comportam ao longo do tempo em um sistema. Ele também registra o comportamento de um único caso de uso do sistema, exibindo os componentes e as mensagens passadas entre esses componentes no diagrama de caso de uso.

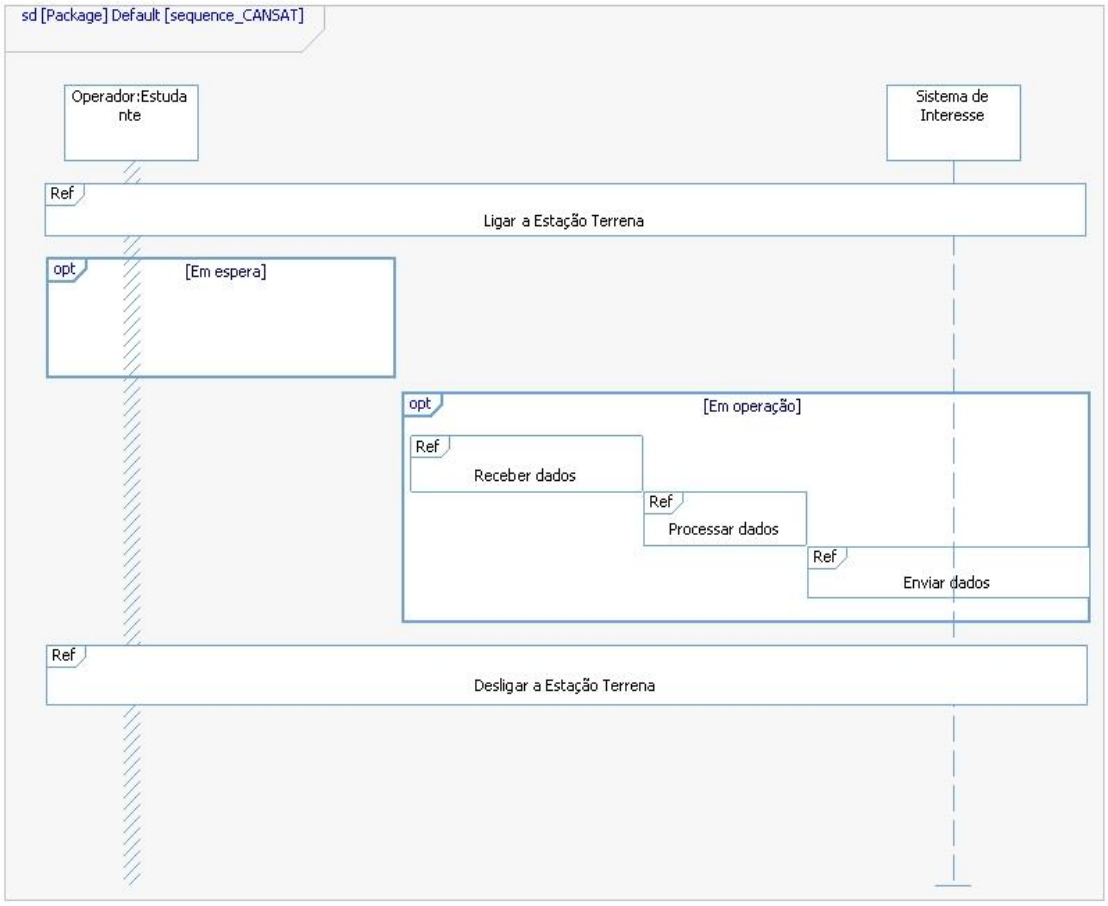


Figura 8 – Diagrama de sequência da linguagem SysML.

Fonte: Flausino, 2013.

8.5.4.3. Diagrama de máquina de estado

O diagrama de máquina de estado denota o comportamento do sistema como uma sequência de estados de um elemento de interação em relação a resposta de eventos, demonstrando a visão dinâmica, sendo importante na modelagem comportamental das interfaces do sistema.

Ele representa o estado ou a situação em que um evento pode encontrar-se no decorrer da execução do processo de um sistema complexo. Com isso, o evento transforma-se de um estado inicial para um estado final através de transição.

Há softwares de modelagem, que permitem animações, simulações e gerações de códigos fontes a partir do próprio modelo desse diagrama.

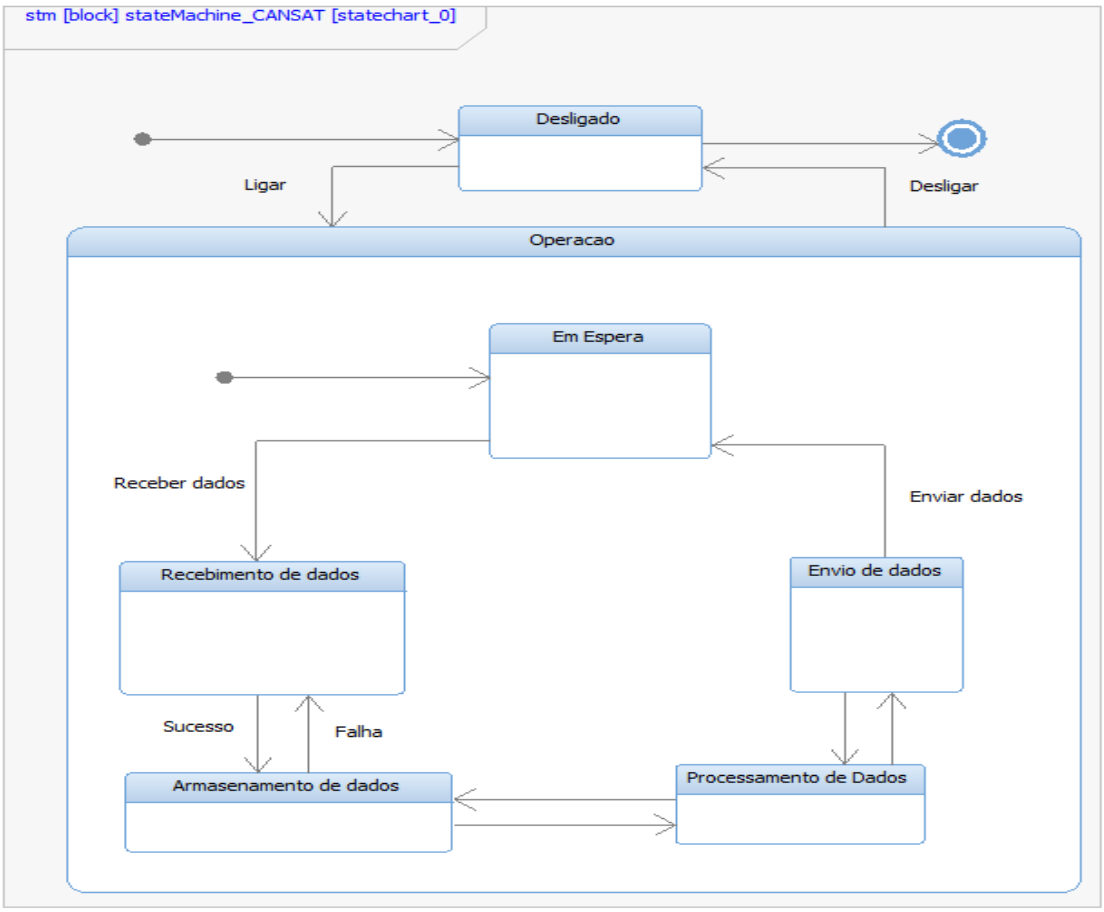


Figura 9 – Diagrama de máquina de estado da linguagem SysML.

Fonte: Flausino, 2013.

8.5.4.4. Diagrama de caso de uso

O diagrama de caso de uso demonstra os requisitos funcionais, os quais são significativos para os *stakeholders*, pois descreve as funcionalidades propostas para um sistema, o qual será projetado, podendo estender um caso de uso como seu próprio comportamento.

Ele também apresenta uma visão externa de como os elementos de um sistema podem estar relacionados, expondo como um sistema se comporta e como esse comportamento será implementado, o que facilita a organização e a modelagem comportamental do sistema. Esse diagrama modela aspectos dinâmicos do sistema, envolvendo a modelagem de contexto, de requisitos e do comportamento desses elementos do ambiente.

Segundo Ivar Jacobson, o diagrama de caso de uso trata-se de um documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para realizar um processo, por exemplo, uma tarefa a ser executada pelo ator no sistema, *login* ou registro em um sistema.

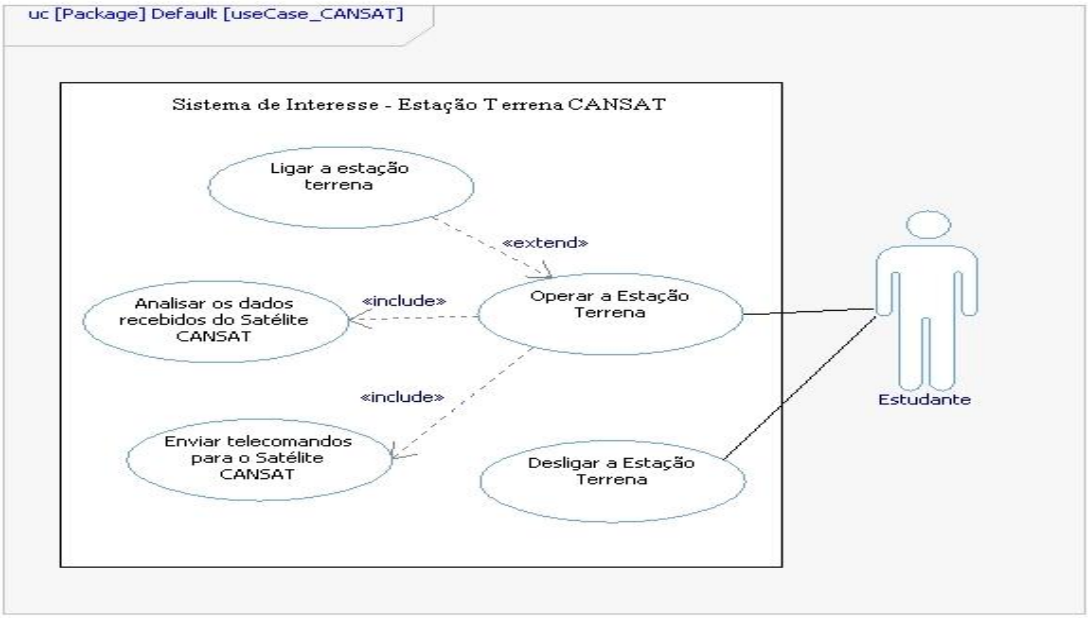


Figura 10 – Diagrama de caso de uso da linguagem SysML.

Fonte: Flausino, 2013.

8.5.4.5. Diagrama de definição de bloco

O diagrama de definição de bloco aponta a estrutura do sistema, expondo seus subsistemas e seus componentes juntamente com suas propriedades, operações e relacionamentos, ajudando na análise e no *design* de sistemas.

Esse diagrama descreve as relações entre cada subsistema, bem como a troca do fluxo de material, energia e informação.

Ele difere dos fluxogramas por representar pequenas partes de um grande sistema com foco no processo lógico, uma vez que, modela a visão estática e estrutural do sistema, mostrando os conjuntos de blocos e seus relacionamentos no nível mais alto de abstração.

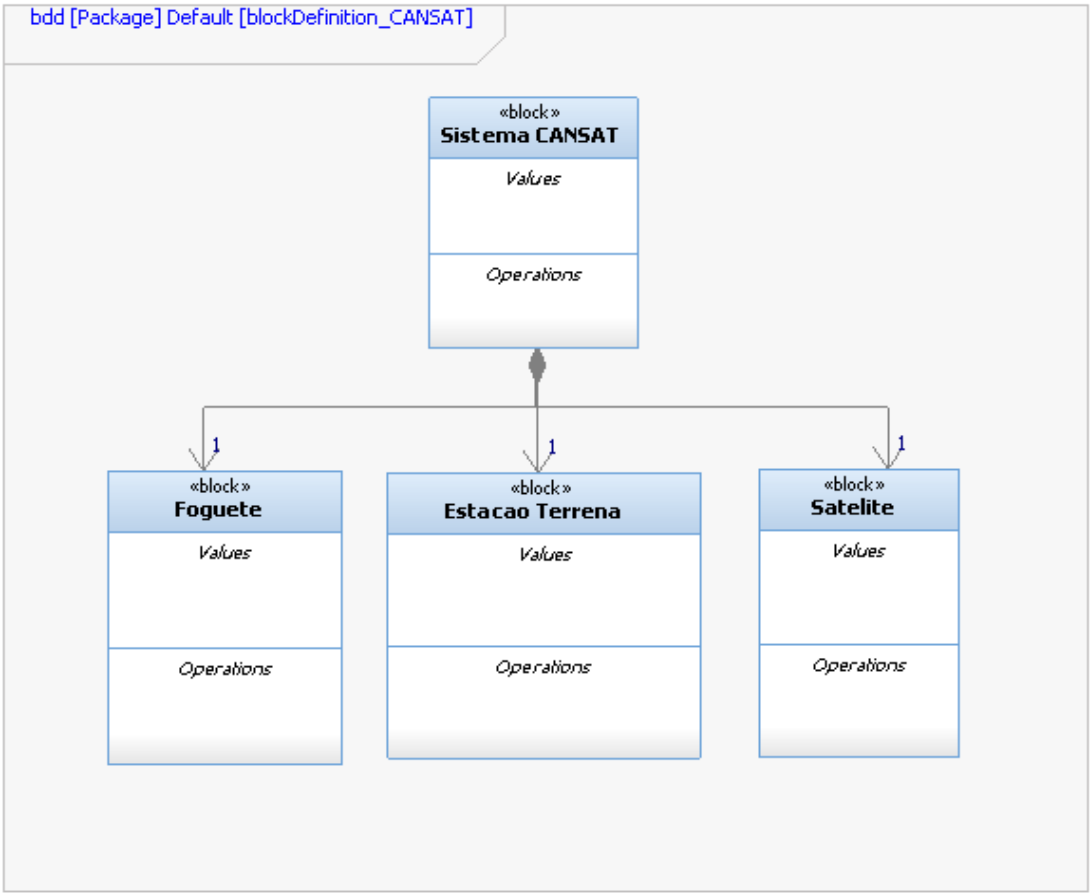


Figura 11 – Diagrama de definição de bloco da linguagem SysML.

Fonte: Flausino, 2013.

8.5.4.6. Diagrama de bloco interno

O diagrama de bloco interno reflete as estruturas internas de seus subsistemas e seus componentes, incluindo suas partes e conectores. Da mesma forma que o diagrama de definição de blocos, também é útil para análise e *design* de sistemas. Já que, se trata da subdivisão do diagrama de definição de blocos.

Esse diagrama descreve as relações entre cada subsistema e a troca do fluxo de material, energia e informação.

Ele modela a visão estática de um processo ou de um sistema, mostrando os conjuntos de blocos e seus relacionamentos, no nível mais detalhado do projeto, ou seja, o nível mais baixo de abstração, pois representa a modelagem gráfica estrutural, possibilitando a modelagem dos componentes de um sistema.

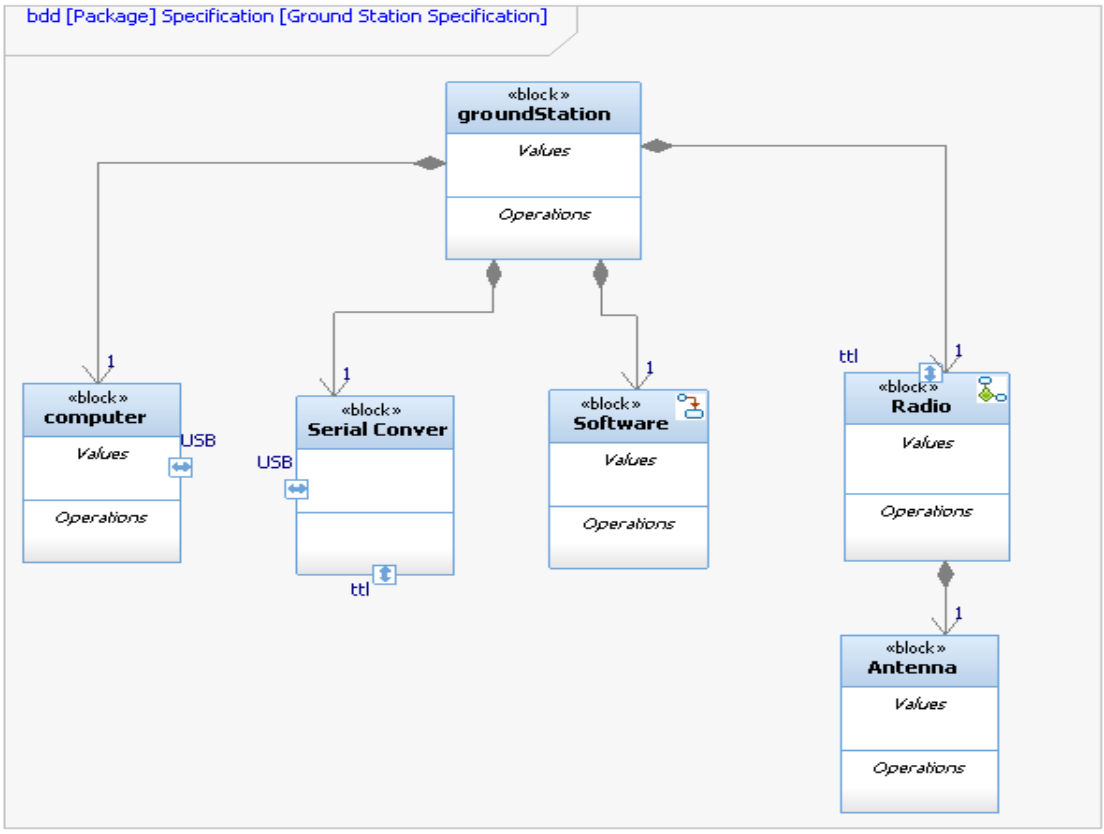


Figura 12 – Diagrama de bloco interno da linguagem SysML.

Fonte: Flausino, 2013.

8.5.4.7. Diagrama de pacote

O diagrama de pacote expõe como um modelo está organizado em pastas, ou seja, o mesmo sistema pode ser modelado de diferentes pontos de vista. Entende-se por pastas, as pastas de um Sistema Operacional (documentos, imagens, musicas, vídeos) para organizar o sistema.

Ele é usado especialmente para ilustrar a arquitetura de um sistema, podendo ser usado em qualquer fase do processo de modelagem, por isso é empregado para gerenciamento dos modelos de um sistema complexo.

Esse diagrama trata-se de pastas que se relacionam com outras pastas através de uma relação de dependência, ou seja, pacotes representam os subsistemas do sistema divididos em agrupamentos lógicos, os quais mostram as dependências entre eles.

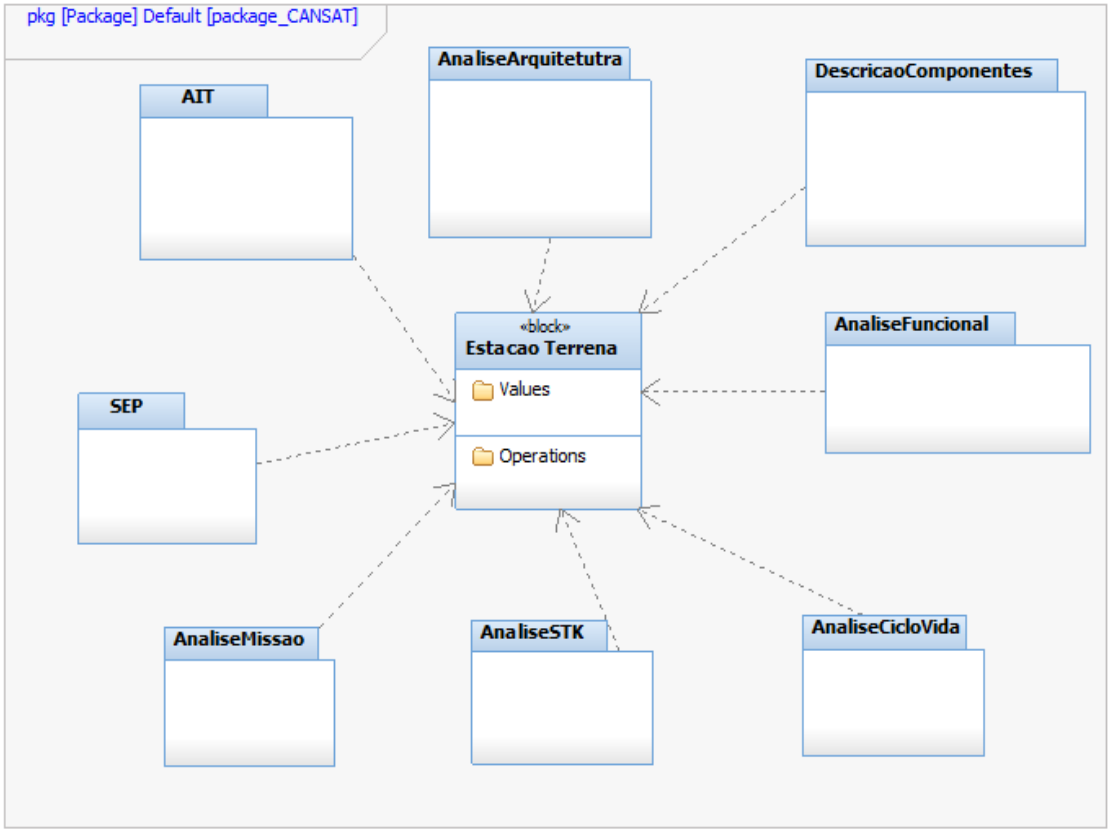


Figura 13 - Diagrama de pacote da linguagem SysML.

Fonte: Flausino, 2013.

8.5.4.8. Diagrama de parâmetro

O diagrama de parâmetro exibe as restrições paramétricas existente entre os elementos estruturais de um sistema. Por isso, pode ser utilizado para medir o desempenho e também realizar a análise quantitativa de um sistema.

Desta forma, ele está relacionado com as equações que modelam o comportamento do sistema, as quais podem ser simuladas pelo *software MATLab/Simulink*.

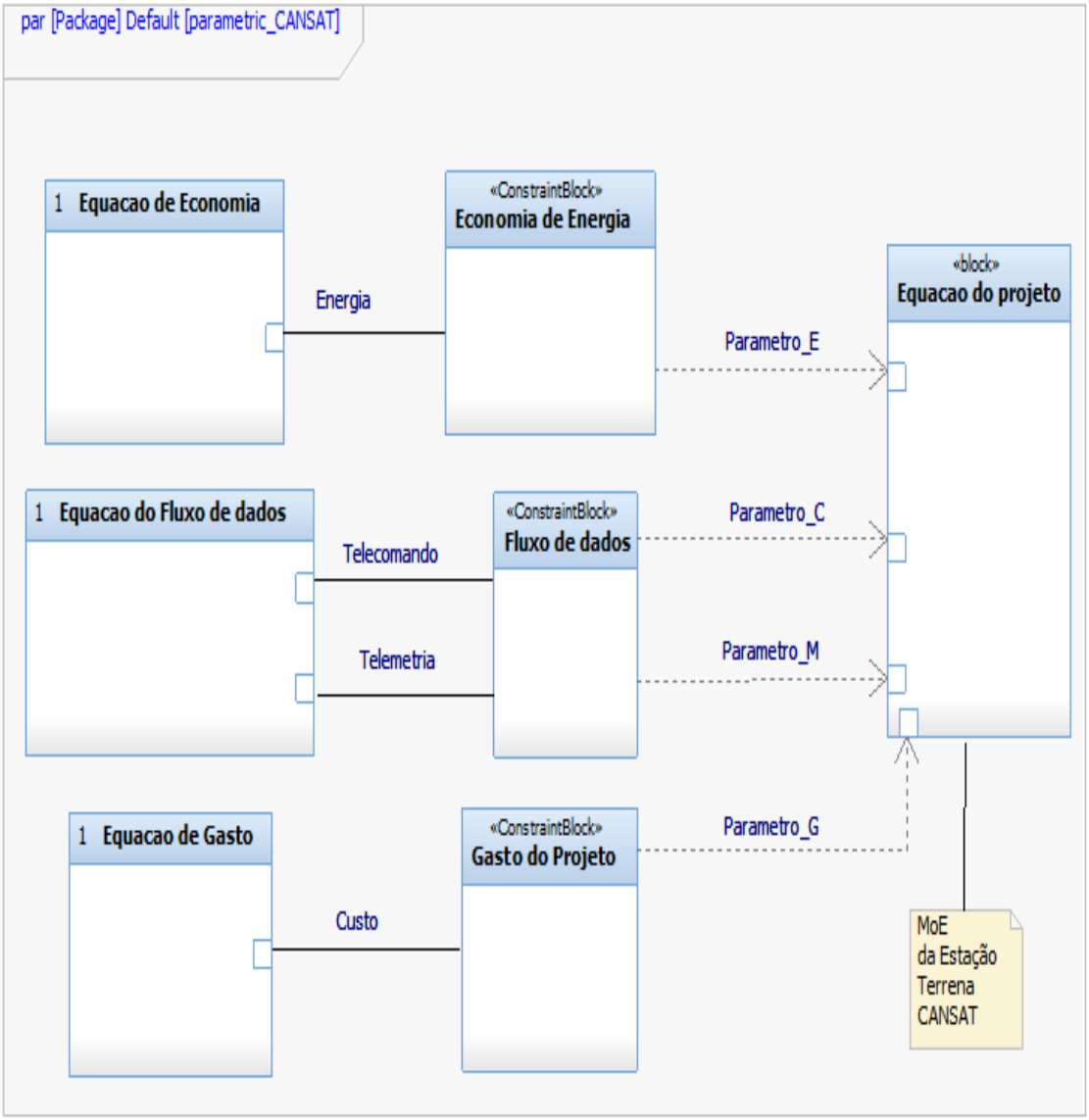


Figura 14 - Diagrama de parâmetro da linguagem SysML.

Fonte: Flausino, 2013.

8.5.4.9. Diagrama de requisito

O diagrama de requisito evidencia as relações entre os requisitos de um sistema, bem como suas relações com os outros elementos de um sistema. Por isso, é uma técnica indispensável, que dá suporte à Engenharia de Requisitos.

Esse diagrama refere-se a um tipo específico do diagrama de definição de bloco.

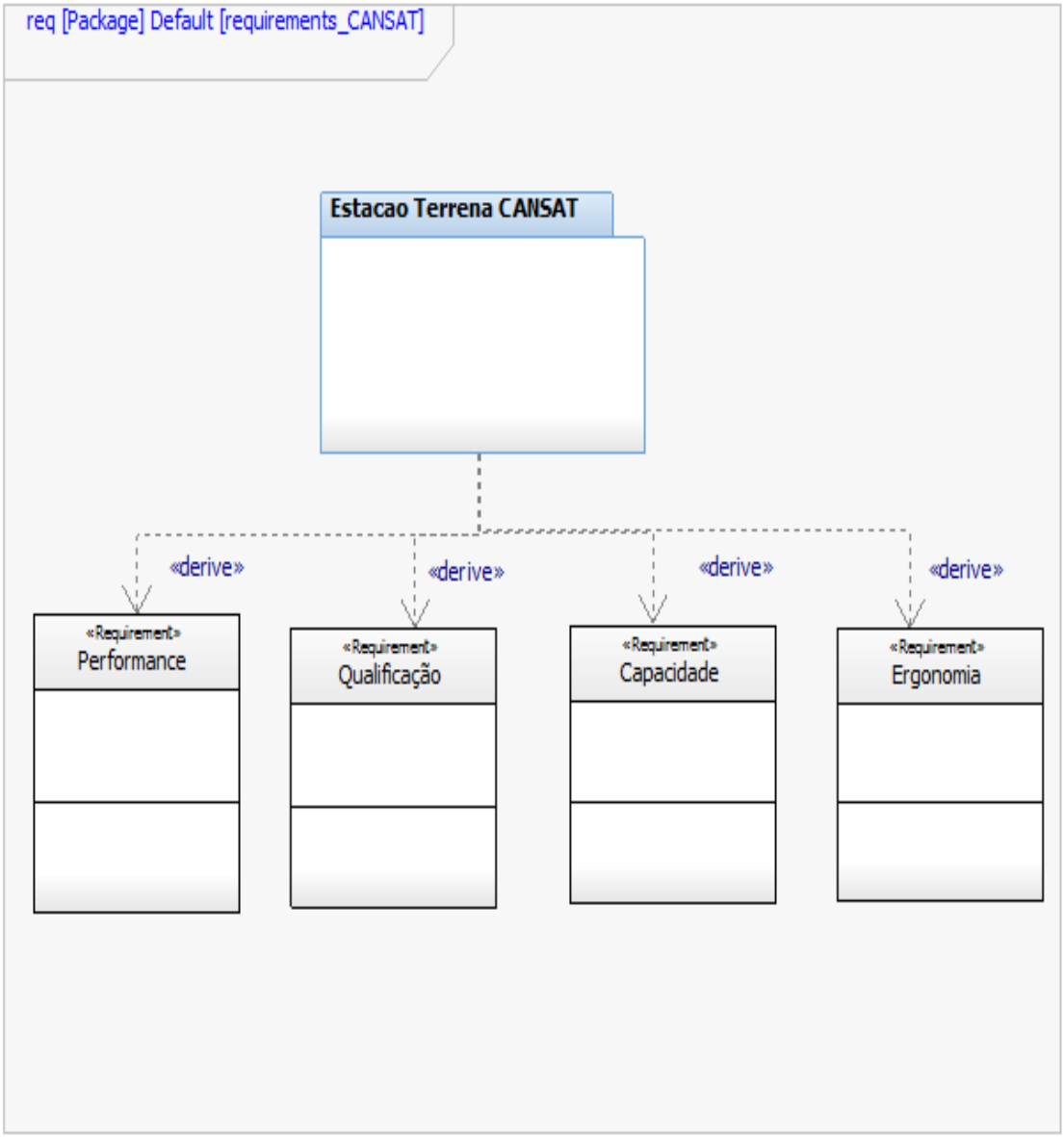


Figura 15 – Diagrama de requisito da linguagem SysML.

Fonte: Flausino, 2013.

8.6. Ferramentas computacionais

As ferramentas computacionais que dão suporte a Engenharia Simultânea de Sistemas são fundamentais para o desenvolvimento de um produto complexo. Em especial os *softwares* de modelagem, pois somente com o apoio deles é possível: a) Realizar o controle de versões dos documentos gerados em cada fase desse processo. b) Corrigir possíveis erros na modelagem. c) Rastrear requisitos, para que qualquer mudança no projeto seja relatado o que será afetado direta ou indiretamente com as ações realizadas.

Quanto mais complexo um produto maior a necessidade de um *software* que auxilie o andamento do processo de Engenharia Simultânea de Sistemas durante o projeto. Devido a essa situação surgiram várias ferramentas para auxiliar a aplicação dos conceitos de Engenharia Simultânea de Sistemas.

Em relação aos *softwares* que dão suporte a linguagem SysML, existe uma questão importante de ser detalhada. Apesar da linguagem SysML permitir a criação de novos tipos de diagramas e de relações, cada *software* possui sua própria característica de modelagem, ou seja, há um certo grau de liberdade em relação ao que se pode e que não se pode fazer na modelagem, que varia em relação a ferramenta adotada para realizar a modelagem do projeto.

Diante do apresentado, segue a descrição de algumas ferramentas que auxiliam o processo de Engenharia Simultânea de Sistemas.

8.6.1. *Cradle*

O *software Cradle*, desenvolvido pela empresa 3SL, ajuda na implementação de um ambiente de Engenharia de Sistemas, permitindo a gestão de requisitos, outras de suas funções são: a) Enumeração e definição de requisitos. b) Análise de sistemas e design. c) Definição da arquitetura de sistemas. d) Modelagem de processos de negócios e engenharia. e) Definição e entrega da capacidade de serviço.

Com esta ferramenta é possível realizar alterações na configuração, através da adição de pacotes de extensão, para maximizar o controle, a eficiência, a segurança e a durabilidade dos projetos, garantindo o orçamento e o tempo previsto para o projeto.

Esse programa possibilita que mais de uma pessoa trabalhe simultaneamente no mesmo projeto, a fim de criar, atualizar e gerenciar colaborativamente todos e quaisquer tipos de informação envolvida em produtos complexos.

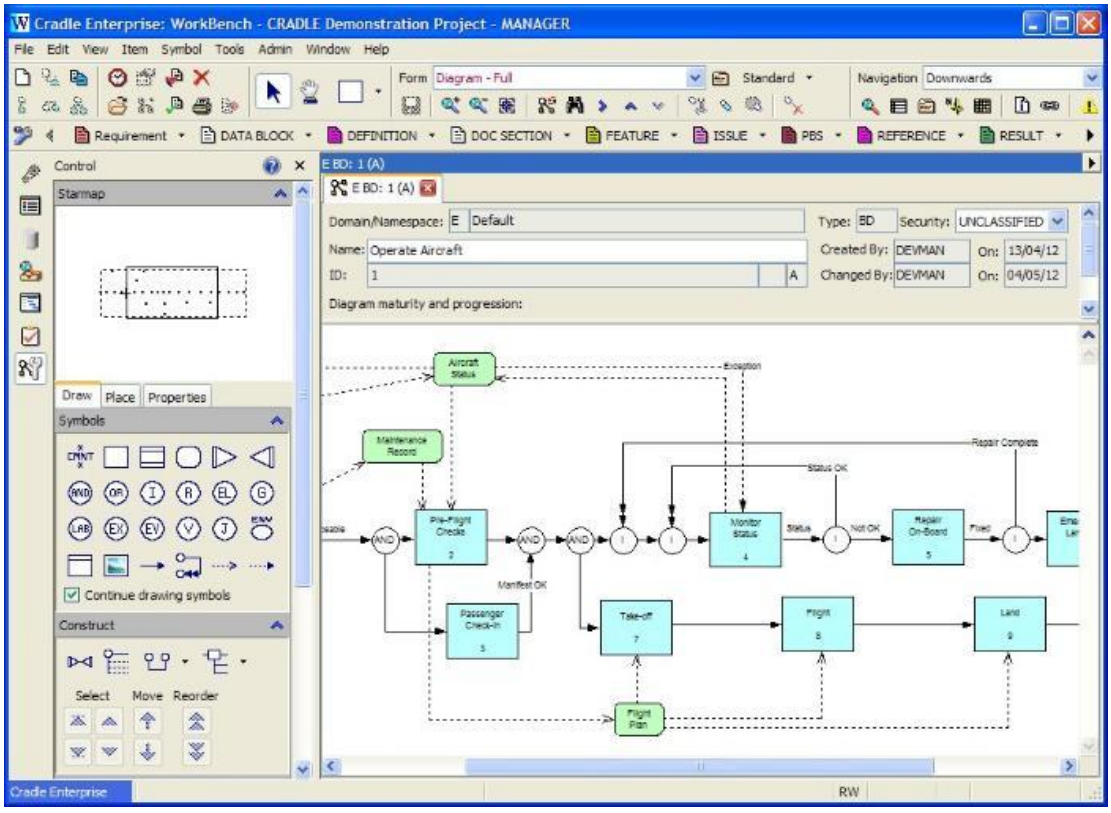


Figura 16 - Tela do *software Cradle.*

Fonte: *Site* da empresa 3SL, 2013.

8.6.2. *Enterprise Architect*

O *software Enterprise Architect*, desenvolvido pela empresa *Sparx Systems*, juntamente com o *plug-in* de extensão para a linguagem SysML é capaz de modelar, projetar e construir um sistema usando a linguagem SysML. A empresa oferece o *trial* dessa ferramenta em seu *site*, sendo possível usá-la por trinta dias.

A partir dos modelos elaborados nesta feramente é possível gerar códigos automaticamente em Java, C#, C++, VB.NET, VB, Python e DLL. Desta forma, a modelagem contribui para que a execução do projeto seja mais rápida, já que se derivam os códigos fontes diretamente dos modelos gráficos. Além disso, é possível também criar relatórios, gerenciar e simular testes, entre outros recursos disponíveis no *software*.

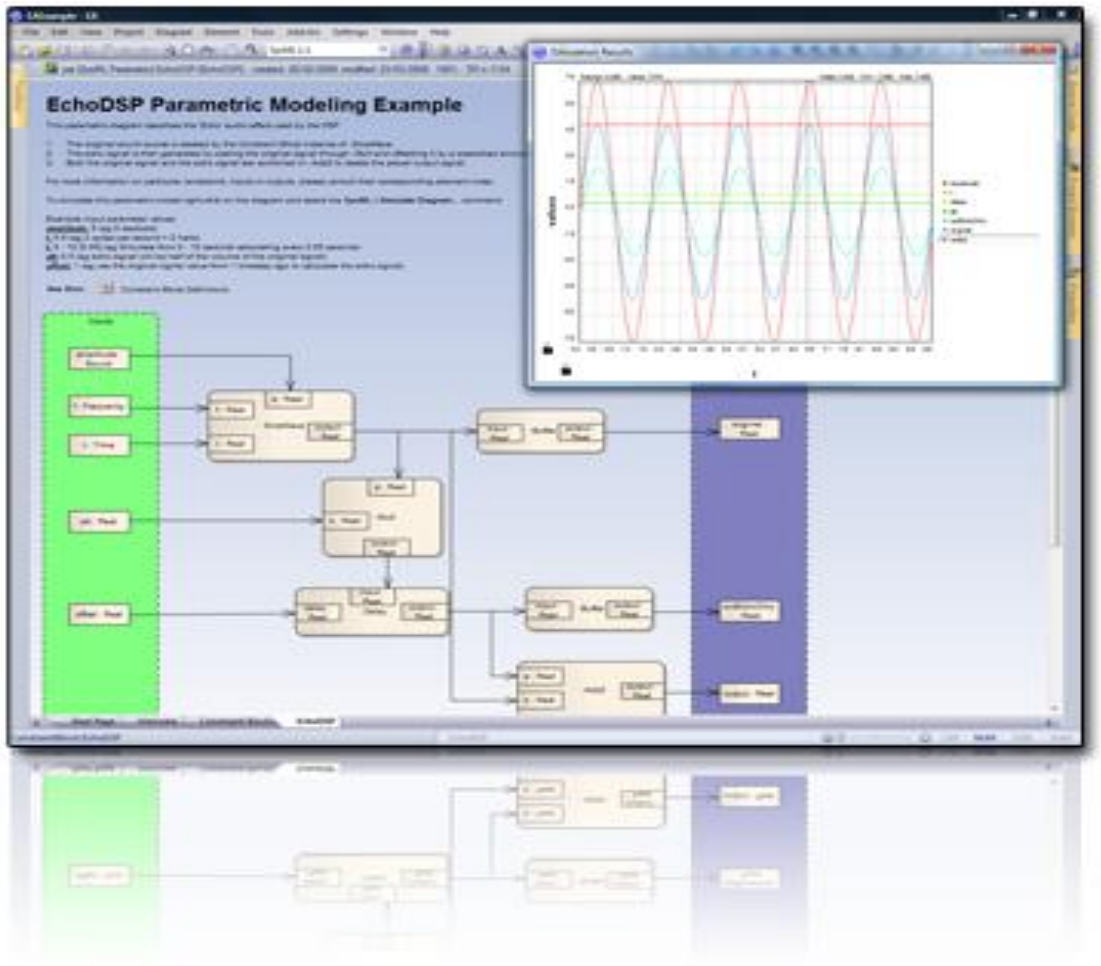


Figura 17 – Tela do *software Enterprise Architect.*

Fonte: *Site* da empresa *Sparx Systems*, 2013.

8.6.3. *IBM Rational DOORS*

O *software IBM Rational DOORS* permite que a comunicação entre os requisitos e os modelos seja eficiente, colaborando para a validação dos requisitos com os *stakeholders*, bem como possibilita a entrega de produtos com maior qualidade e em menor tempo.

Essa ferramenta também é capaz de capturar, relacionar, analisar e atualizar os requisitos, ajudando as organizações responsáveis pelo desenvolvimento de produtos complexos a obterem a satisfação dos *stakeholders*.

O *IBM Rational DOORS* importa e exporta em XMI e em XML e gera código em linguagem C, C++, JAVA e documentos automaticamente.

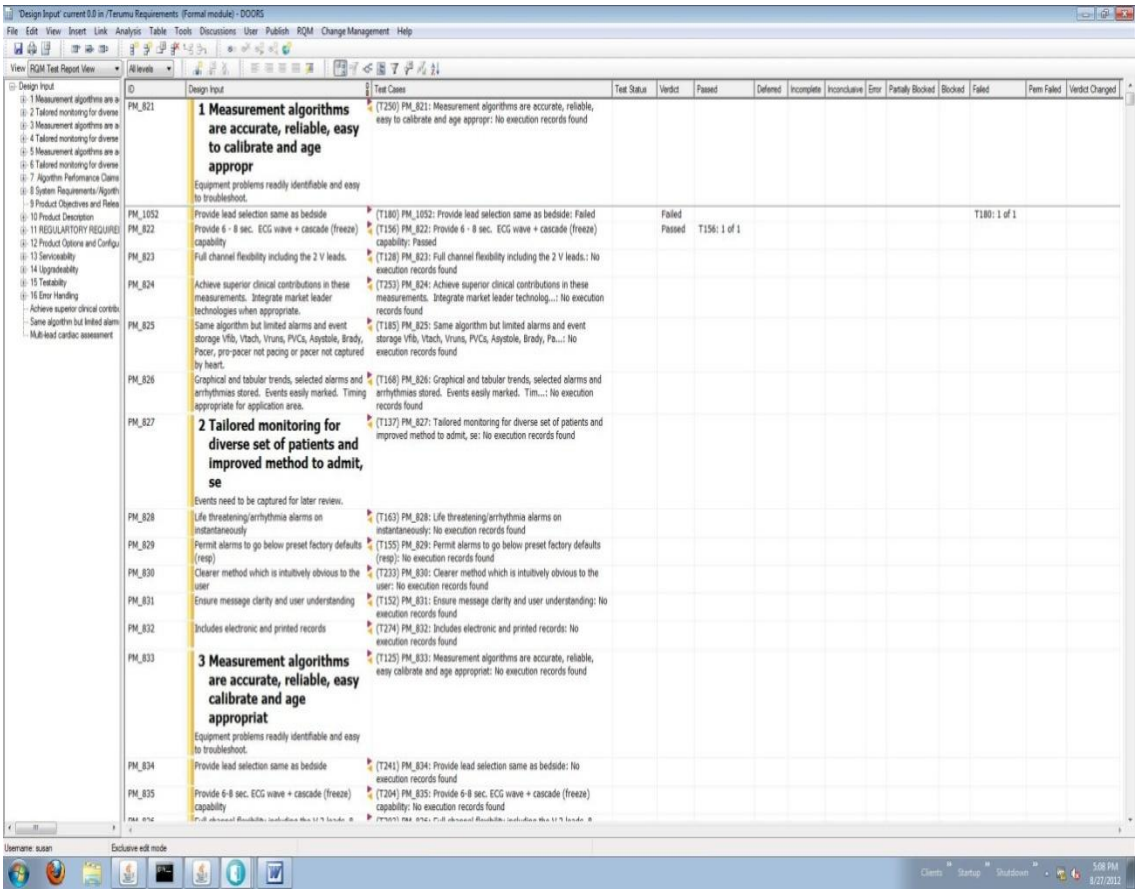


Figura 18 – Tela do *software IBM Rational DOORS.*

Fonte: *Site* da empresa IBM, 2013.

8.6.4. *IBM Rational Rhapsody*

A ferramenta *IBM Rational Rhapsody* possui varias funcionalidades, as quais atendem as etapas do método proposto por Loureiro (2010), pois suporta análise, *design*, desenvolvimento, testes e simulações, sendo que para realizar simulações é necessária a integração desta com o *software MatLAB/Simulink*, o qual extrai os dados do diagrama de parâmetro da linguagem SysML para realizar as simulações matemáticas.

Esse *software* modela em diversas linguagens gráficas, exporta e importa em XMI e em XML e também é possível gerar códigos nas linguagens C, C++, JAVA e documentos automaticamente. A sua interface gráfica é prática e fácil de ser manipulada pelo usuário.

O *IBM Rational Rhapsody* possibilita também a rastreabilidade e a consistência entre os diagramas de requisito da linguagem SysML e os requisitos armazenados no banco de dados do *IBM Rational DOORS*, através da integração desses programas, uma vez que os dois são da mesma empresa, a IBM. No site da empresa é possível, após fazer o cadastro, realizar o *download* da licença *trial* por trinta dias.

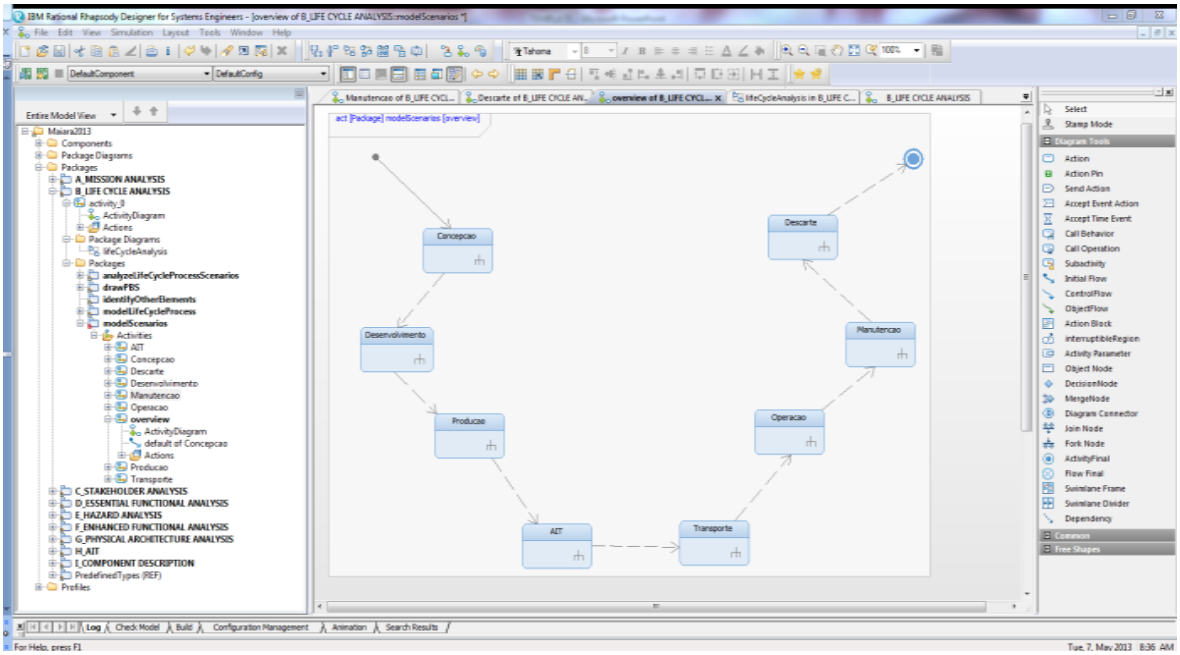


Figura 19 - Tela do *software IBM Rational Rhapsody*.

Fonte: Flausino, 2013.

8.6.5. *MagicDraw - SysML Plug-in*

O *software MagicDraw* em conjunto com o *SysML Plug-in* permite a modelagem de requisitos através do diagrama de requisito da linguagem SysML, sendo que esses são armazenados em um banco de dados interno do programa, para que testes e simulações validem esses requisitos. Esse diferencial auxilia o desenvolvimento do sistema complexo, detalhando o projeto, por meio de análises e validações automaticamente.

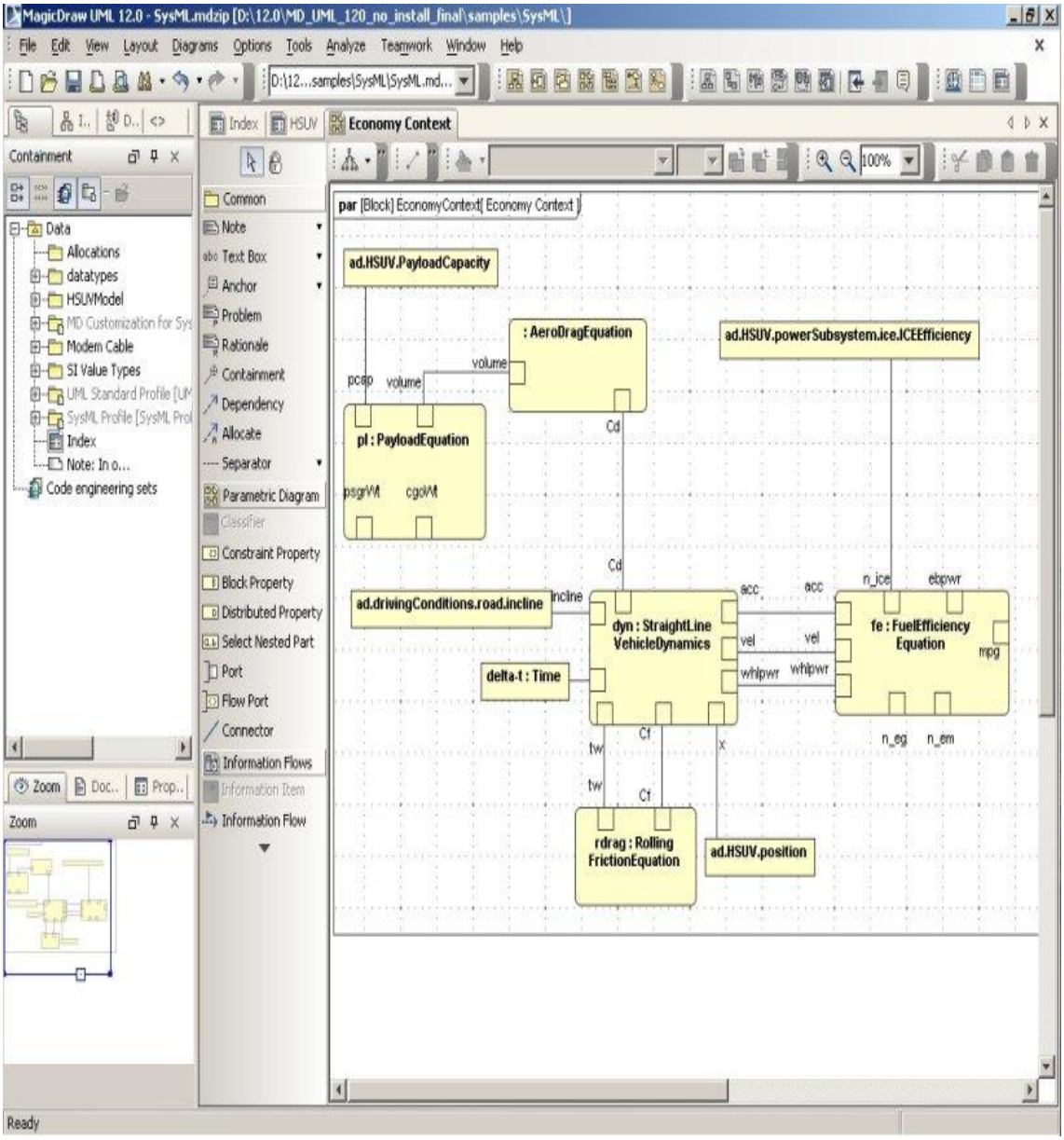


Figura 20 – Tela do *software MagicDraw*.

Fonte: *Site* da empresa *MagicDraw*, 2013.

8.6.6. *Modelio System Architecture*

O programa de modelagem gráfica *Modelio System Architecture* possui suporte para as linguagens de modelagem padrão, atualmente, como, *BPMN*, *UML*, *MDA*, *SysML*, *TOGAF*.

No *site* da empresa *ModelioSoft* é possível encontrar a licença *trial* dessa ferramenta, por dez dias, porém esta versão não oferece o diagrama de requisitos da linguagem SysML.

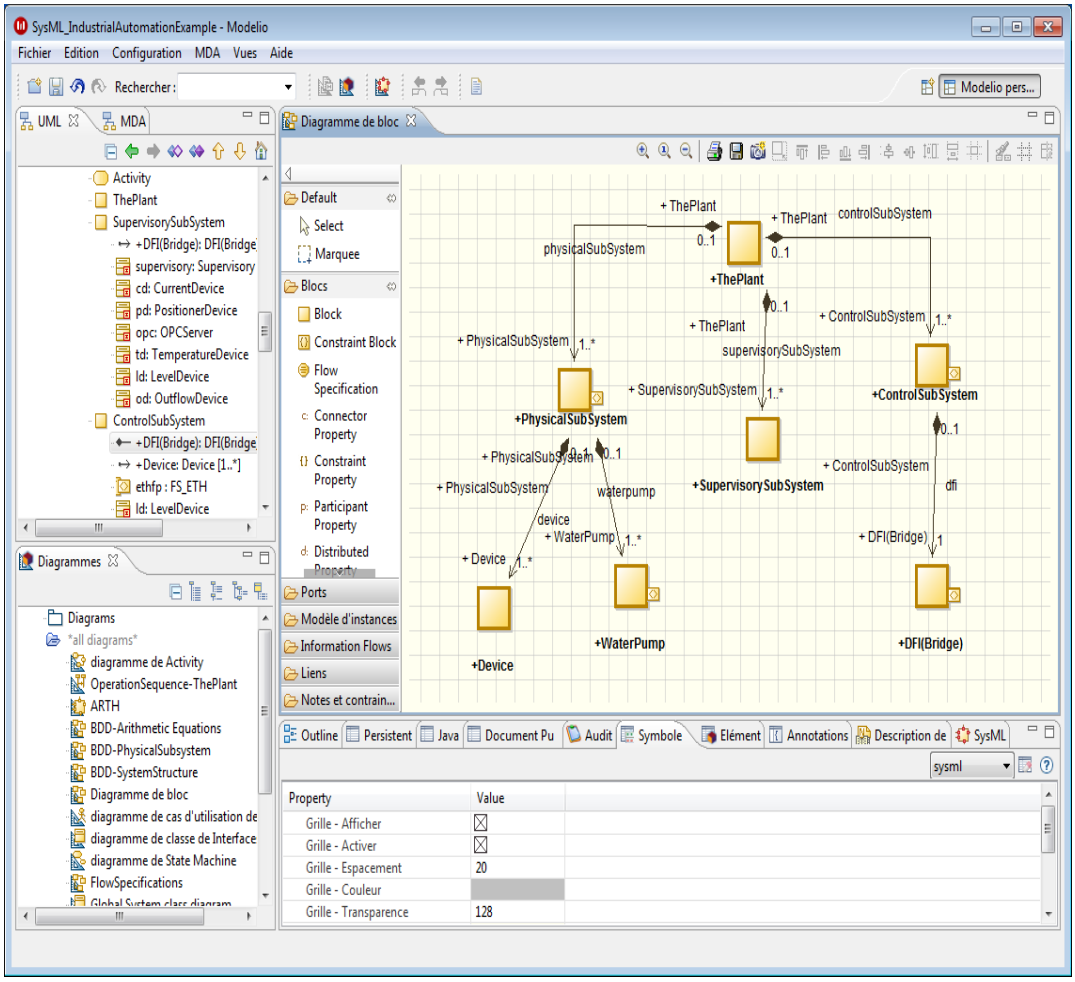


Figura 21 – Tela do *software Modelio System Architecture*.

Fonte: *Site* da empresa *ModelioSoft*, 2013.

8.6.7. *Papyrus UML for SysML*

O *Papyrus UML for* SysML é uma ferramenta *open source*, a qual proporciona ao usuário a modelagem na linguagem SysML, implementado a maioria dos estereótipos e propriedades relacionadas a essa linguagem gráfica, segundo a especificação SysML V1.1, por isso, apresenta limitações em relações as outras ferramentas computacionais apresentadas anteriormente neste trabalho, uma vez que, a linguagem SysML encontra-se na versão 1.3.

Essa ferramenta trata-se de um dos projetos do *Eclipse*, para dar suporte aos modelos das linguagens UML e SysML.

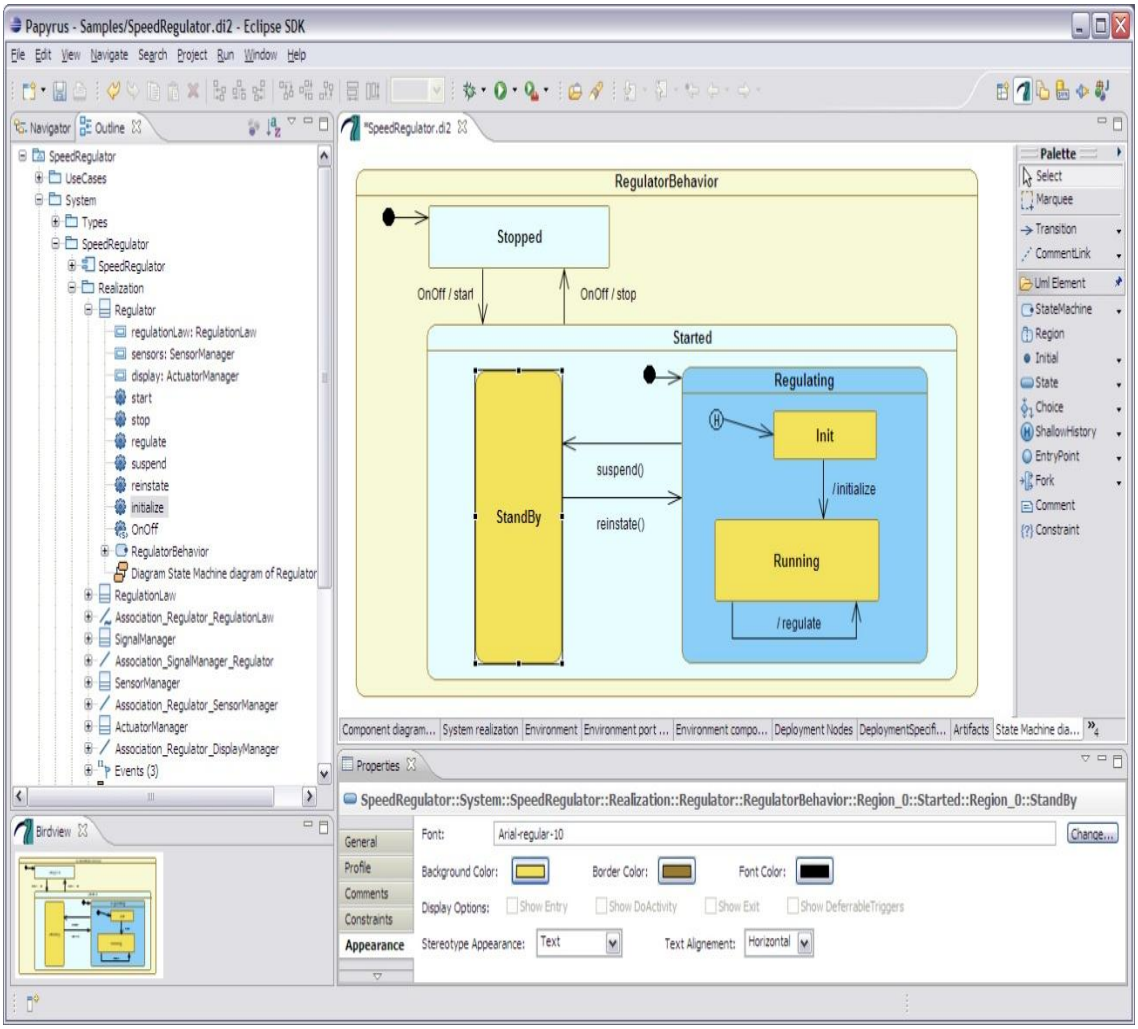


Figura 22 – Tela do *software Papyrus UML for SysML*.

Fonte: *Site* da comunidade *Eclipse*, 2013.

8.6.8. *Top Cased - SysML*

O *software TopCased - SysML* também é uma ferramenta *open source*, que possibilita a modelagem em linguagem SysML. A sua plataforma baseia-se no *Galileo* (*Eclipse* 3.5) e requer uma JVM 1.5 (*Java Virtual Machine*) instalada, o seu *download* pode ser feito como um aplicativo autônomo ou pode ser instalado diretamente através do gerenciador de instalação do *Eclipse*.

Primeiramente, ele foi projetado para atender tanto a análise critica de *softwares* como a de *hardwares*. Entretanto, ainda não possui todas as funções necessárias para desempenhar essa tarefa, atualmente encontra-se em desenvolvimento. Igualmente a ferramenta *Papyrus UML for SysML* também apresenta restrições no desenvolvimento dos modelos na linguagem SysML.

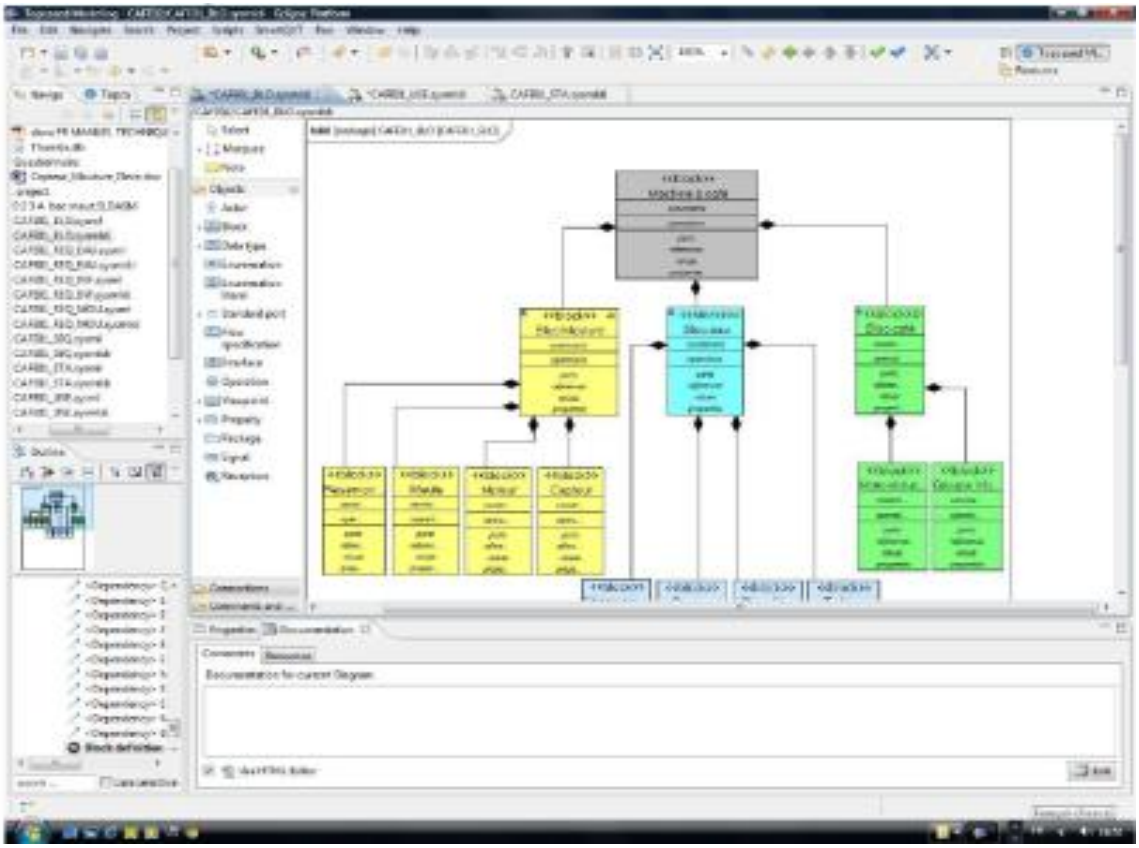


Figura 23 – Tela do *software TopCased – SysML*.
Fonte: *Site* da comunidade *TopCased – SysML*, 2013.

8.7. Sistema CANSAT

O sistema CANSAT é composto por um foguete, um picossatelite e uma estação terrena. Esse produto espacial foi o exemplo utilizado para desenvolver o *template* deste trabalho, dando foco principalmente ao seguimento da estação terrena. Esse produto espacial foi escolhido, pois foi o trabalho usado em sala de aula por Loureiro (2011).

O picossatelite do sistema CANSAT, chama se CANSAT, o nome CANSAT deriva-se literalmente do que é composto o picossatelite, ou seja, satélite de lata, que vem do inglês, *can* significa lata. A seguir uma ilustração desse sistema, Figura 24.

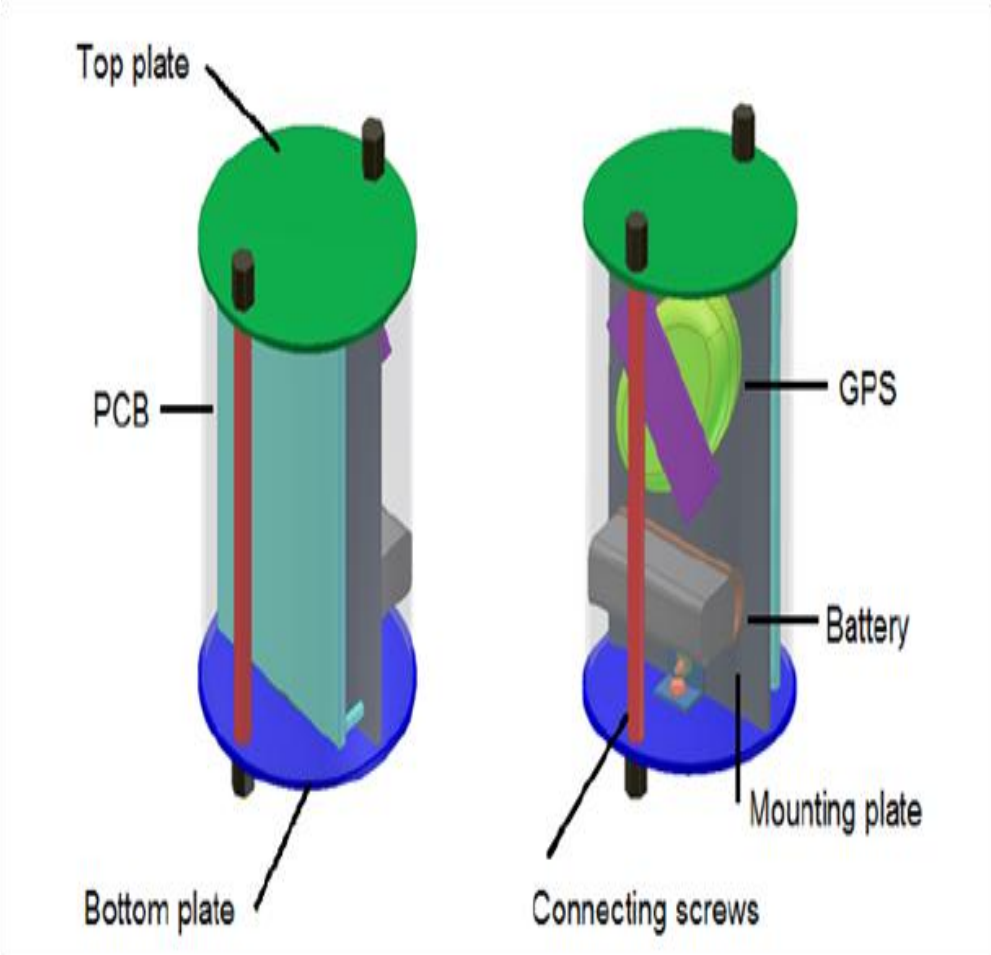


Figura 24 – Estrutura do picossatelite do sistema CANSAT.
Fonte: Documento do projeto do sistema CANSAT, 2011.

Esse sistema tem como objetivo a simulação de um satélite verdadeiro, que é lançado por um foguete e depois de lançado deve de forma autônoma capturar alguns dados, como, localização geográfica, aceleração, temperatura, pressão e umidade, os quais deverão ser enviados para a estação terrena, a qual deve enviar telecomandos e receber telemetrias do picossatelite.

Desta forma, a estação terrena deve ser capaz de interpretar os dados recebidos. A missão do seu lançamento é a realização das missões de cada equipe, bem como a sua aterrissagem em segurança.

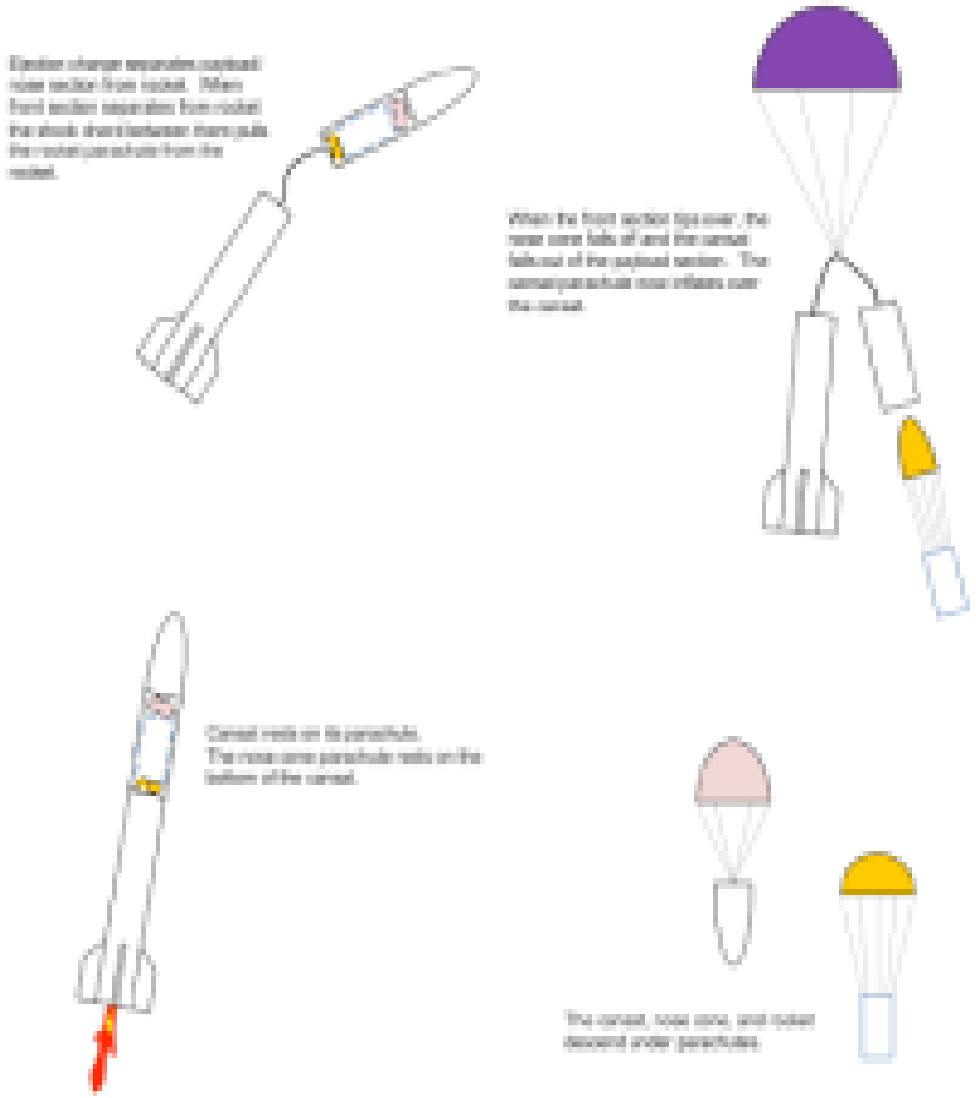


Figura 25 – Lançamento do picossatelite do sistema CANSAT.

Fonte: Documento do projeto do sistema CANSAT, 2011.

A Figura 26, a seguir, mostra as interfaces existentes entre os subsistemas do sistema CANSAT, ou seja, a troca de material, energia e informação.

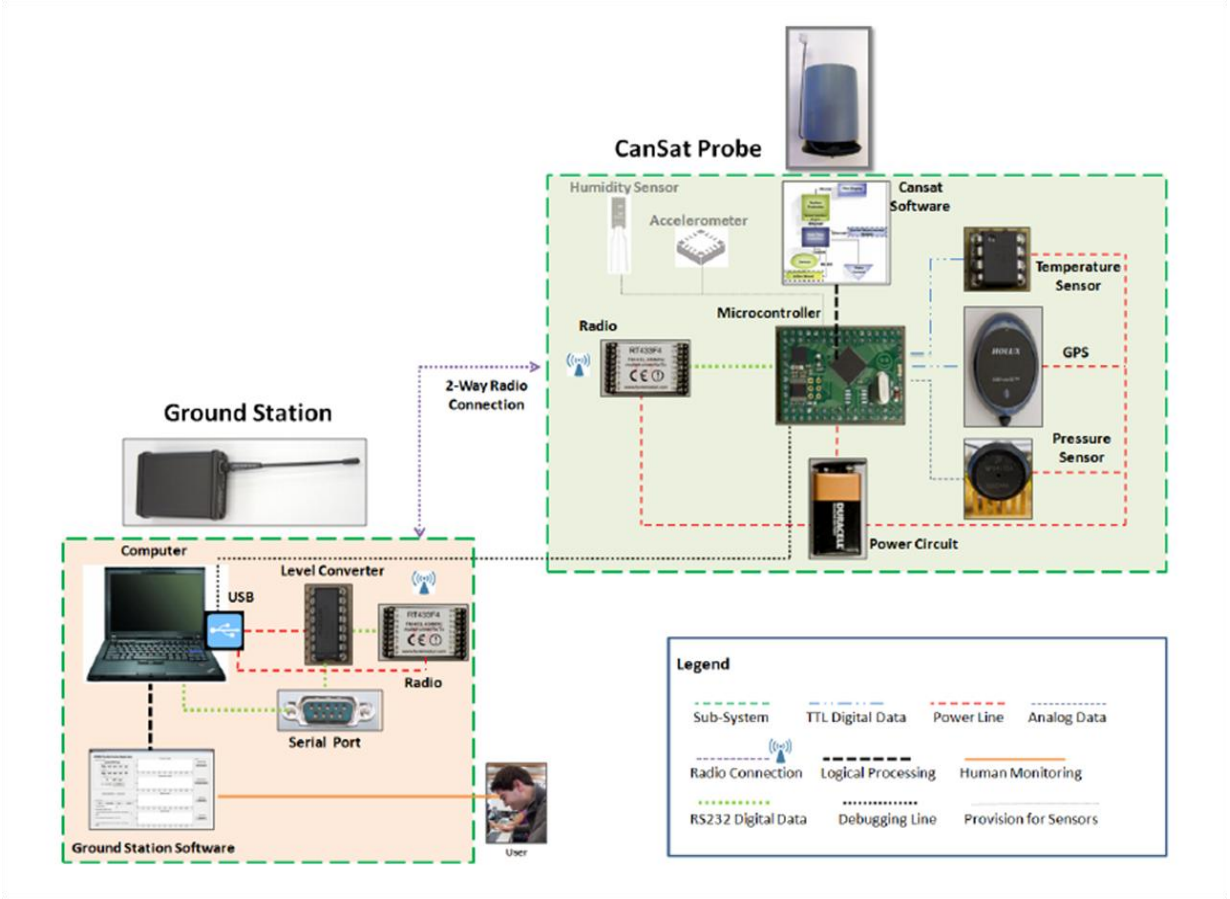


Figura 26 – Interfaces do sistema CANSAT.

Fonte: Documento do projeto do sistema CANSAT, 2011.

A seguir, serão apresentados diagramas de definição de bloco da linguagem SysML, que mostram a estrutura do sistema CANSAT.

Na Figura 27, observam-se os mesmo elementos da Figura 26 modelados no diagrama de definição de bloco, sendo que o diagrama contém uma grande vantagem em relação a uma simples figura, a geração de código fonte automaticamente, isso é possível através da ferramenta *IBM Rational Rhapsody*.

Essa prática vem obtendo ganhos significativos ao substituir, a documentação de sistemas complexos, páginas descritivas, por modelos estruturados, cuja utilização consiste em um dos objetivos deste trabalho.

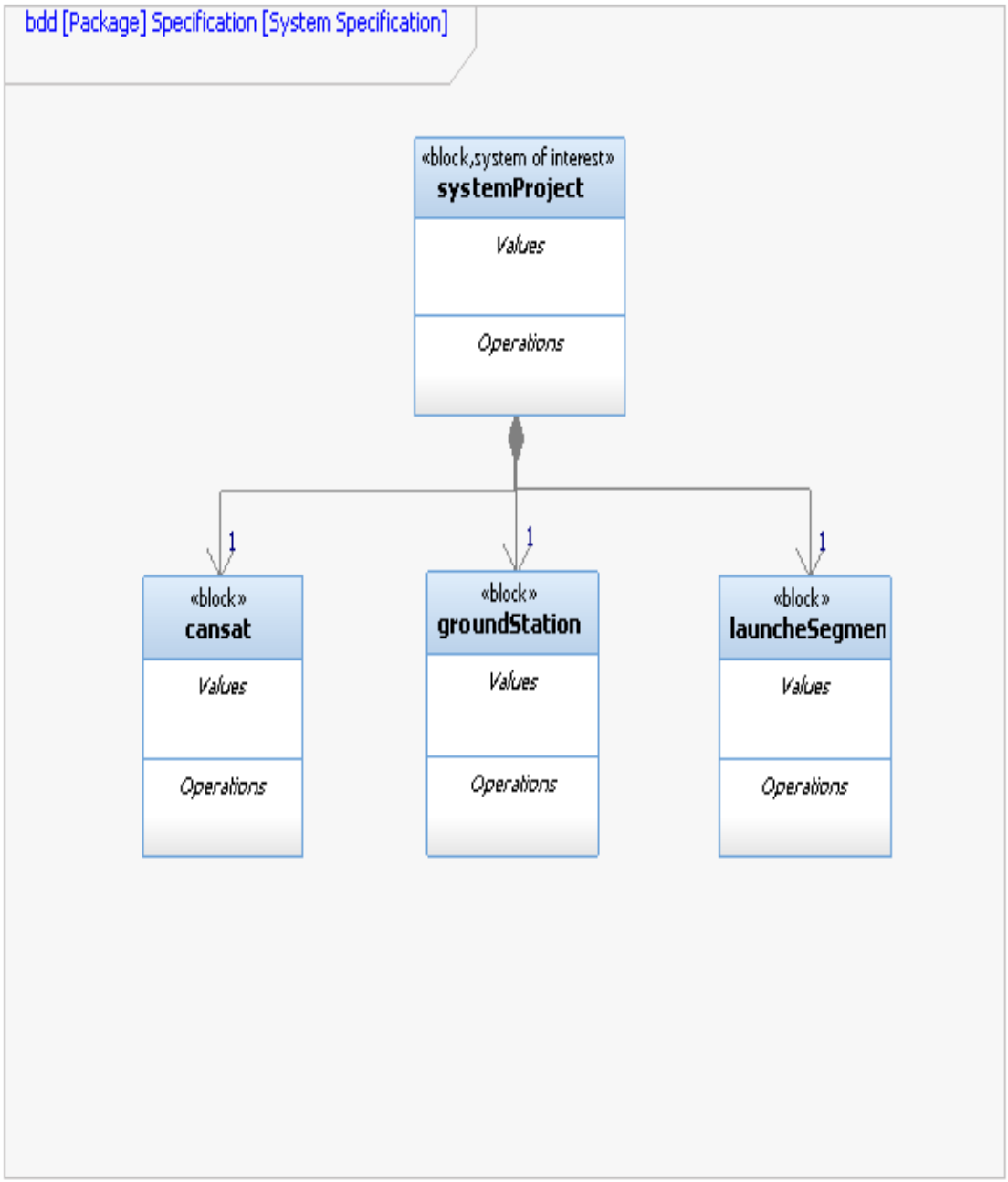


Figura 27 – Diagrama de definição de bloco do sistema CANSAT.

Fonte: Flausino, 2013.

No próximo diagrama, Figura 28, foi dado um *zoom* no bloco da estação terrena, mostrando os subsistemas pertinentes a ela, também foi usado um diagrama de definição de bloco, pois esse permite que seja expresso as partes e os relacionamentos dos subsistemas.

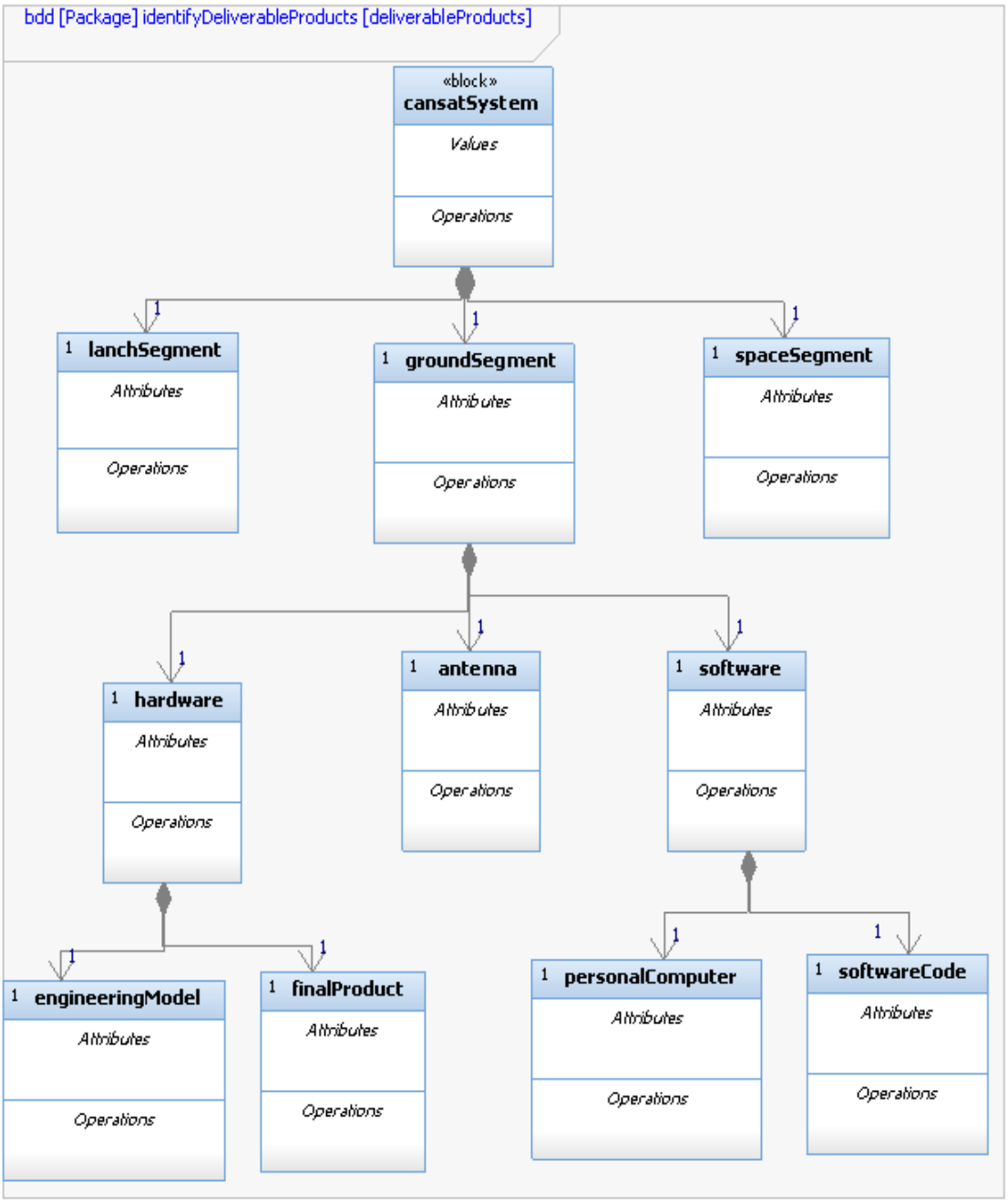


Figura 28 – Diagrama de definição de bloco da estação terrestre (1).

Fonte: Flausino, 2013.

No diagrama a seguir, Figura 29, há um nível de detalhamento maior do que os diagramas anteriores, demonstrando que é importante usar uma ferramenta computacional para modelar sistemas complexos, pois o desdobramento de um sistema pode ser muito grande, uma vez que, o nível de detalhamento pode chegar a especificar os componentes, ou seja, mostrar todo o conjunto de interfaces de um sistema.

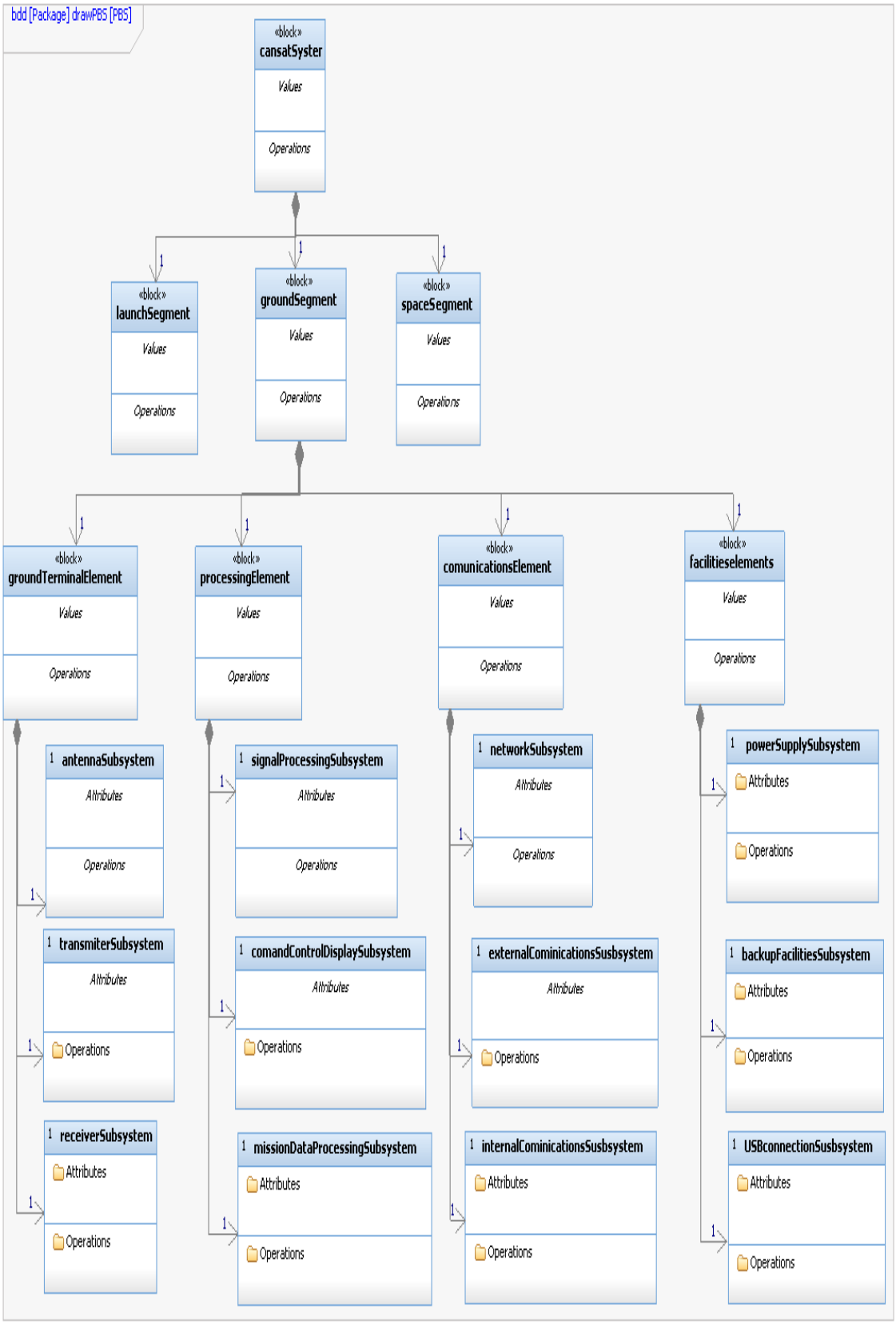


Figura 29 – Diagrama de definição de bloco da estação terrestre (2).

Fonte: Flausino, 2013.

9. DESENVOLVIMENTO

Sabe-se que, a modelagem, a simulação e os testes são imprescindíveis nos dias de hoje, para que não haja desperdícios de recursos, como, o financeiro, o tempo e a mão de obra especializada.

Assim, a linguagem SysML foi criada para dar suporte durante todo o ciclo de vida de um produto complexo, usando a Engenharia Simultânea de Sistemas. A Figura 30 apresenta o ciclo de vida em "V", muito utilizado nos processos de Engenharia de Sistemas.

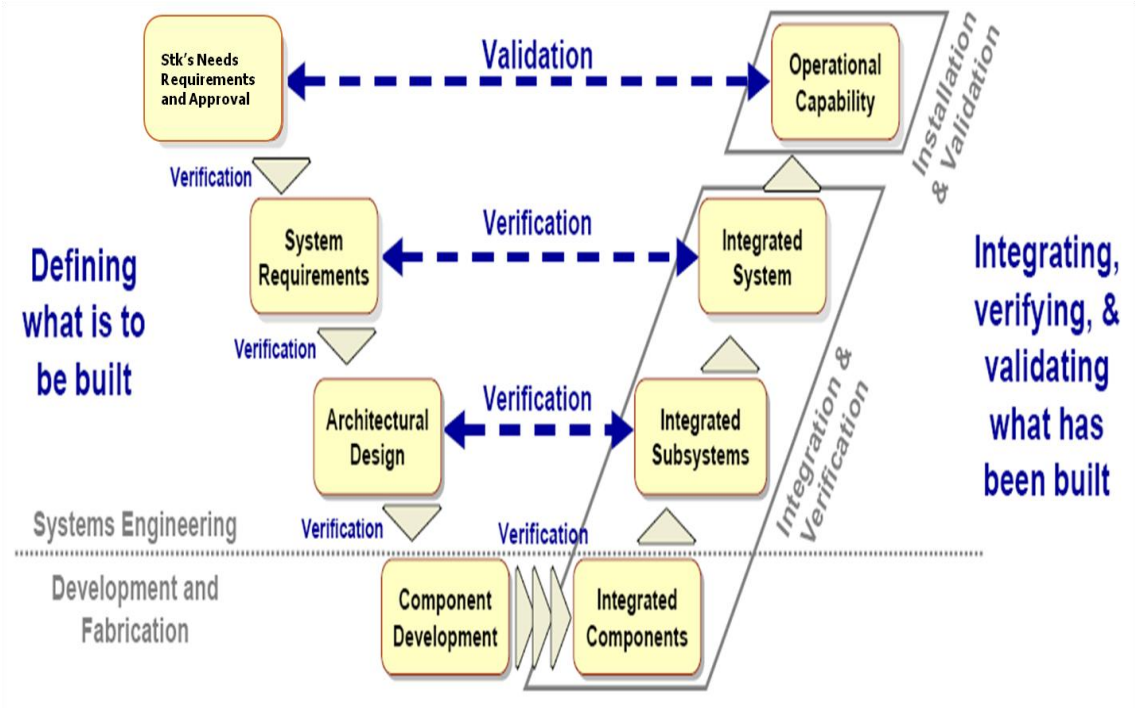


Figura 30 – Ciclo de vida em "V".

Fonte: Loureiro, 2011.

Durante o estudo deste trabalho, verificou que a linguagem SysML possui uma sintaxe e uma semântica muito bem definida, mas aceita que sejam criados novos estereótipos de diagramas e de relações entre os elementos do modelo. E devido a essa característica da linguagem, existe a garantia de não haver ambiguidade na construção de modelos.

Desta forma, há um grande ganho da utilização na linguagem SysML, a possibilidade de transformação no arquivamento de informações imprescindíveis para um projeto, pois ela permitiu que os documentos descritivos sejam agora simulados e analisados, tanto comportamental como estruturalmente, antes de serem implementados de modo definitivo.

Isso é possível, pois houve a facilitação ao acesso de dados e de modelos, através da rastreabilidade de requisitos de um sistema.

Diante desse contexto, este trabalho propõe o uso da linguagem SysML no desenvolvimento de um *template* a partir do método desenvolvido por Loureiro (1999), o qual pode ser consultado no ANEXO B, sendo assim, através dessa tabela descritiva do método foram feitos estudos e análises para saber quais etapas seriam possíveis de modelar usando a linguagem SysML.

O ANEXO B é uma tabela, que foi detalhada por Loureiro (2011), na tentativa de esclarecer as etapas de seu método. E a forma como foi apresentada, em tabela *Excel*, não permite que as entradas e saídas sejam observadas, bem como as suas interações existentes entre elas. Por isso, foi necessária a criação deste *template*.

Para modelar o *template* foi utilizado o *software IBM Rational Rhapsody* 7.6. e o sistema escolhido para ser modelado foi o sistema CANSAT, exemplo de sala de aula, o qual não abrangeu a organização do produto.

O primeiro passo, para a construção do *template* foi analisar o método de Loureiro (1999) e em seguida escolher um exemplo para aplicação desse método. Assim, como resultado dessas análises e da escolha do exemplo foi feita a seguinte Figura 31, para sintetizar o entendimento desta pesquisa.

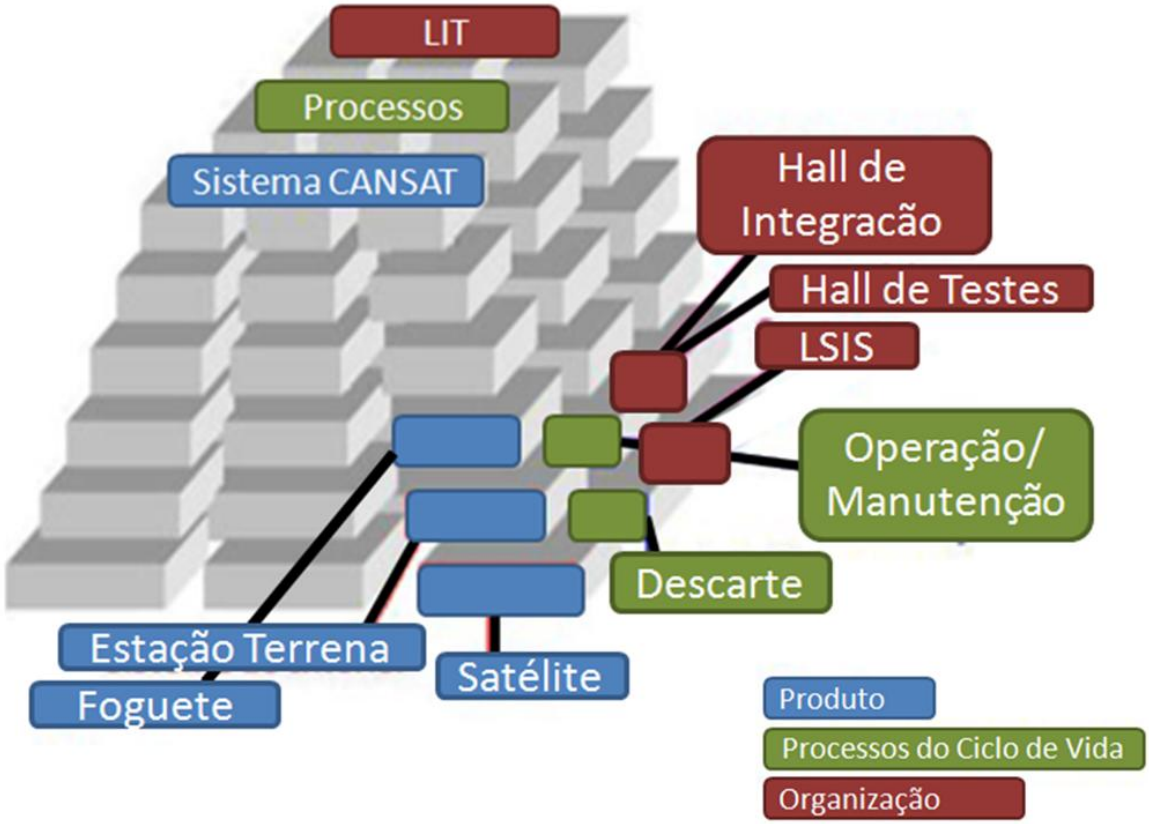


Figura 31 – Aplicação do Método de Loureiro.

Fonte: Flausino, 2013.

A partir disso, foi desenvolvida uma nova tabela, ANEXO C, a qual apresenta a ordem lógica em que as etapas do método ocorrem, ou seja, este trabalho propõe a organização da tabela proposta por Loureiro (2011), ANEXO B, bem como a especificação do uso da linguagem SysML, última coluna da tabela. A colona denominada “saída” foi desenvolvida por Mayoral (2013).

Com base nas análises e estudos sobre o método de Loureiro (2011) foi possível identificar que as etapas que podem ser modeladas usando a linguagem SysML. Afinal essa linguagem é um suporte gráfico que visa expressar a estrutura e o comportamento de um sistema complexo.

Para o desenvolvimento desta pesquisa foi feito também um estudo bibliográfico da linguagem SysML, que proporcionou uma visão de suas potencialidades para a modelagem de sistemas mecânicos, elétricos e espaciais.

A SysML é uma linguagem nova, ainda em desenvolvimento. Por isso, foi necessário participar de fóruns *online* e ler especificações diretas do desenvolvedor para constante atualização e para a criação de novos estereótipos. Pois durante o processo de adaptações para a modelagem do *template*, novos estereótipos foram criados.

Em seguida, foram levantadas as vantagens e as desvantagens de usar a linguagem SysML. A principal vantagem encontrada é o fato de ser uma linguagem padrão, com grande capacidade de desenvolvimento. A desvantagem, que aparentemente existe não cabe especificamente a linguagem SysML, mas sim a todas as linguagens de modelagens, isto é, para modelar um sistema não basta saber só a linguagem, é preciso também ter conhecimento técnico sobre alguma ferramenta computacional que de suporte a essa linguagem de modelagem, além de também conhecer profundamente o sistema a ser modelado.

Durante o desenvolvimento deste trabalho percebeu se que a linguagem SysML como é especificada pelos seus desenvolvedores (OMG e INCOSE) muitas vezes não é a mesma apresentada nas ferramentas computacionais, isto é, cada *software* tem a sua própria customização da linguagem SysML, por exemplo, alguns são mais rígidos quanto a semântica, outros não.

Os modelos foram estudados, pois permitem que o sistema seja melhor compreendido por todos envolvidos no projeto. Pois a modelagem permite: a) Visualizar e controlar um sistema inteiramente. b) Especificar a estrutura/comportamento de um sistema. c) Proporcionar um *template* para construção de um sistema (objetivo deste trabalho). d) Documentar as decisões tomadas em cada fase do projeto. E quanto mais complexo for o sistema, mais importante se torna a sua modelagem. e) Realizar simplificações e reaproveitamento nos sistemas a fim de gerenciar os riscos.

Para modelar este *template* foi importante pensar em cada nível do sistema, sendo fundamental que esse pensamento seja o mais abstrato possível, isto é, o sistema deve ser pensado de maneira funcional, por exemplo, muitas vezes o cliente pede um manômetro, mas o que ele realmente precisa é de um simples objeto que meça a pressão do seu sistema.

O pensamento sistêmico ajuda na analise funcional, uma vez que, o todo sempre depende das relações entre as partes, sendo a parte algo indivisível. Isso ajuda a olhar para o mesmo sistema e enxergar diferentes pontos de vista, para entender completamente o seu funcionamento.

A seguir será discutido como em cada etapa foi possível utilizar a linguagem SysML.

Para desenvolver este *template* foi necessário utilizar os digramas estruturais, pois demonstram como o sistema está organizado e quais são seus subsistemas, suas partes, seus componentes, sendo assim o responsável por organizar as informações e os modelos, facilitando depois os acessos desejados.

Uma vez que, os blocos representam esses elementos, com o diagrama de definição de bloco é possível visualizar os relacionamentos existentes entre cada elemento de um sistema, como associações e generalizações presentes, sendo possível em cada bloco documentar as suas características, como propriedades, valores, portas, operações e recepções.

A organização dos modelos está diretamente relacionada ao pensamento hierárquico necessário para construção de um sistema complexo e para demonstrar isso foram utilizados os diagramas de pacote da linguagem SysML.

Já o diagrama de bloco interno está diretamente relacionado com o diagrama apresentado anteriormente, o diagrama de definição de bloco, pois esse diagrama apresenta as partes dos sistemas, ou seja, um nível abaixo do modelado pelo diagrama de definição de bloco, sendo possível detalhar a estrutura interna do sistema e as suas relações existentes.

Quanto aos diagramas comportamentais é possível visualizar as entradas e saídas do cada atividade. Um deles é o diagrama de caso de uso, o qual é muito utilizado para validar requisitos com os próprios *stakeholders*, o qual possui uma sintaxe muito simples, de fácil entendimento.

Através dele é possível visualizar as funções que o sistema desempenhará segundo as ações dos atores. As funções são representadas por elipses e os autores não necessariamente precisam ser pessoas, podem ser outros sistemas.

Seguindo esse raciocínio, o diagrama de atividade, que também modela aspectos dinâmicos do sistema, mostra as funcionalidades de um sistema e a interação entre elas, bem como o fluxo de material energia e informação das entradas e saídas. Esse diagrama está relacionado a algum caso de uso do sistema, por exemplo, depositar dinheiro em uma conta.

O diagrama de atividade pode também representar os fluxos conduzidos por processamentos, isto é, um gráfico de fluxo, que mostra o fluxo de controle de uma atividade para outra.

Ainda como diagrama de comportamento tem-se o diagrama de sequência, o qual representa o comportamento de um sistema em ações no tempo, sendo capaz de modelar a sequencia de ações dentro um sistema complexo, ou seja, cada mensagem enviada para uma ação dispara esta ação para que possa ser executada. Entende-se por mensagens os serviços solicitados de uma atividade a outra, e as respostas desenvolvidas para as solicitações

Em relação à inovação da linguagem SysML, a qual é apresentada a seguir, os dois novos tipos de diagramas.

a) O primeiro trata dos parâmetros de um sistema, é a partir desse diagrama que é possível, com o auxilio de uma ferramenta computacional, simular os resultados esperados. O *software* indicado para realizar essas simulações é o *MATLab/Simulink*.
O diagrama de parâmetro é uma exclusividade da linguagem SysML, bem como o próximo, o diagrama de requisito.

b) O segundo diagrama de requisitos, o qual se for modelado com o auxilio de uma ferramenta computacional, como o *IBM Rational Rhapsody* poderá ser integrado com o *IBM Rational DOORS*, que ajudará na rastreabilidade de requisitos já que quanto mais complexo for um sistema mais requisitos a serem satisfeitos serão necessários.

Para o que se espera do desenvolvimento deste *template*, chegou-se à conclusão que para modelar um sistema há mais de uma dimensão a ser considerada, ou seja, o modelador deve conhecer em detalhes a linguagem que será usada na modelagem, a ferramenta na qual a modelagem será realizada e o sistema que será modelado, sem isso, não há "empregabilidade" da linguagem, pois ela perde a sua funcionalidade, que é representar o ambiente de forma fiel.

É importante lembrar que a SysML é apenas uma linguagem. Ela necessita de um "método" que guie a sua aplicação em determinado contexto. Recomenda-se o estudo da linguagem SysML para que o desenvolvedor tenha embasamentos para saber aplicá-la no momento certo. Além disso, é fundamental conhecer a ferramenta computacional que será utilizada para modelar o sistema.

9.1. Diagrama de contexto

Pode-se criar, segundo norma da própria linguagem SysML, novos estereótipos de diagramas, de elementos e de relações, o que faz dessa linguagem não ser prescritiva.

Desta forma, para identificar os *stakeholders* do sistema CANSAT, Figura 32, neste trabalho, foi criado o diagrama de contexto, o qual também serve para representar tanto o contexto funcional, Figura 33, como o contexto físico, ou seja, define o ambiente em que o sistema vai pertencer. Esse novo diagrama foi elaborado a partir do diagrama de definição de bloco.

Foi criado esse diagrama dada a importância dos *stakeholders* no processo de Engenharia Simultânea de Sistemas. Por isso, é importante valorizar as informações que os *stakeholders* oferecem, nas respostas que eles dão nas entrevistas. Isso justifica a importância de se perguntar várias vezes sobre o mesmo assunto, sempre tentando pensar funcionalmente.

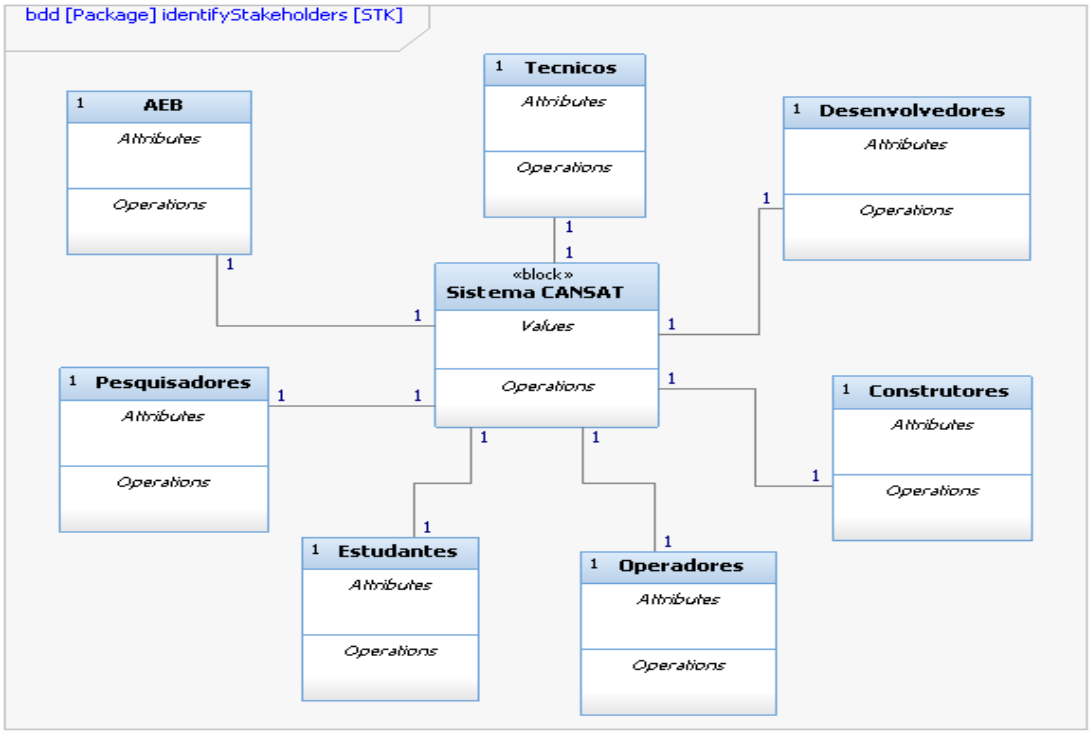


Figura 32 – Diagrama de contexto da linguagem SysML (1).

Fonte: Fonte: Flausino, 2013.

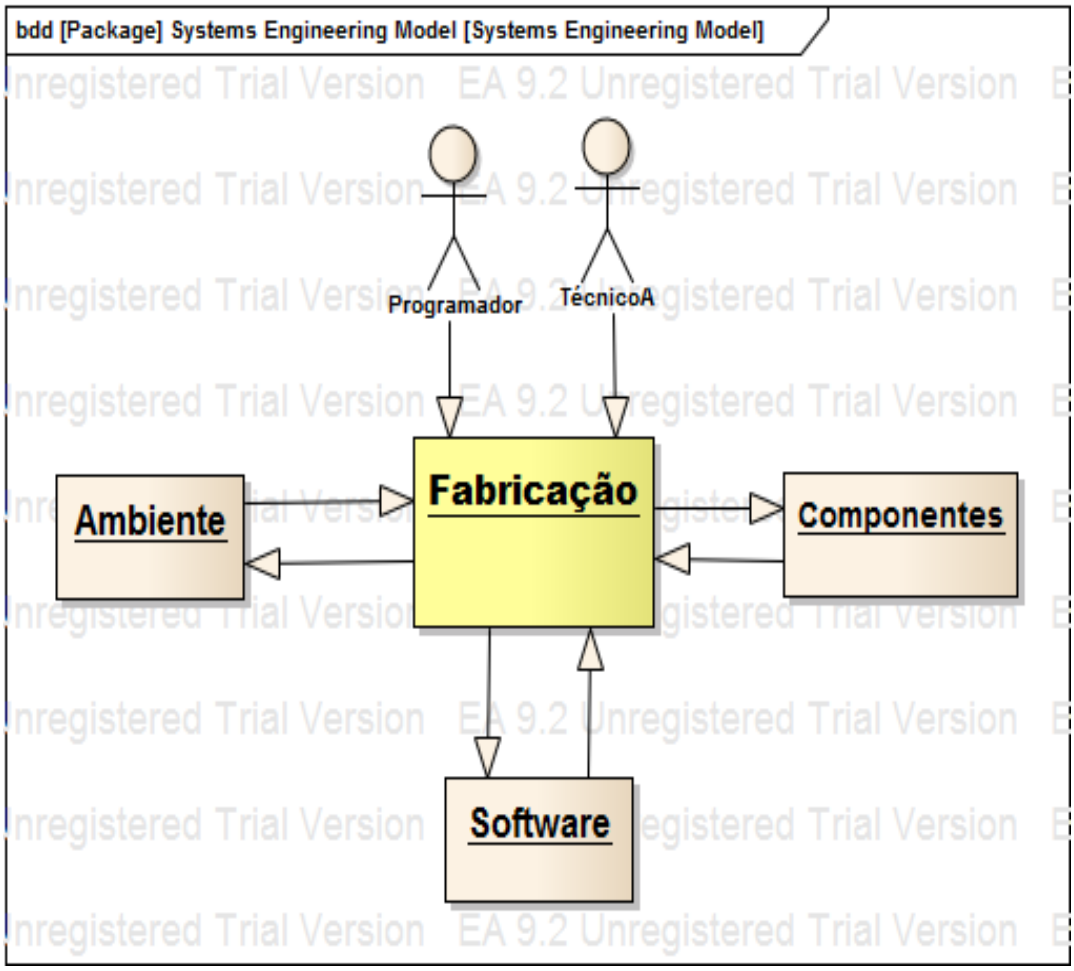


Figura 33 - Diagrama de contexto da linguagem SysML (2).

Fonte: Flausino, 2013.

Seguem algumas características do diagrama de contexto, desenvolvido neste trabalho.

a) Apresenta o conjunto de elementos relacionados para um determinado propósito, como especificar uma operação
b) Define o ambiente em que o sistema vai pertencer.
c) Demonstra as características do sistema.
d) É composto por fluxos de dados que mostram as interfaces entre o sistema e as entidades externas.
e) Permitem identificar os limites dos processos, as áreas envolvidas com o processo e os relacionamentos com outros processos e elementos externos à empresa (ex.: clientes, fornecedores).
f) Representa o objeto do estudo, o projeto, e sua relação ao ambiente.
g) Representa todo o sistema como um único processo.

10. RESULTADO

O objetivo deste *template* é prover melhor entendimento do método proposto por Loureiro (1999), usando a linguagem SysML, para ilustrar como os seus diagramas podem ser aplicados em cada etapa do método, pois com o *template* desenvolvido neste trabalho é possível visualizar o todo de maneira organizada, facilitando a sua análise.

Para se chegar à modelagem proposta neste trabalho levou-se muito tempo, pois sintetizar as idéias é um processo lento e difícil, sendo que construir modelos é fazer uma síntese da realidade, em razão disso o problema deve ser bem estruturado já no início do projeto para assegurar que a modelagem seja fiel à realidade do produto.

A ordem em que os itens estão apresentados, não é necessariamente a ordem que deve ser seguida para aplicar o método em um projeto, mas é a ordem lógica de estruturação para modelagem.

10.1. *Template*

A Figura 34 apresenta todos os pacotes utilizados para modelar o método de Loureiro (1999), os quais estão estruturados na ferramenta *IBM Rational Rhapsody*. Através de estudos críticos sobre esse método, propõe esta organização hierárquica de pacotes da linguagem SysML. Uma vez que, o modelo deve ser estruturado em pacotes para que as etapas sejam facilmente identificadas.

O *template* necessitou especialmente desse tipo de diagrama, pois ele tem a mesma função das pastas dos Sistemas Operacionais, que guardam informações relacionadas entre si, ajudando na organização do sistema.

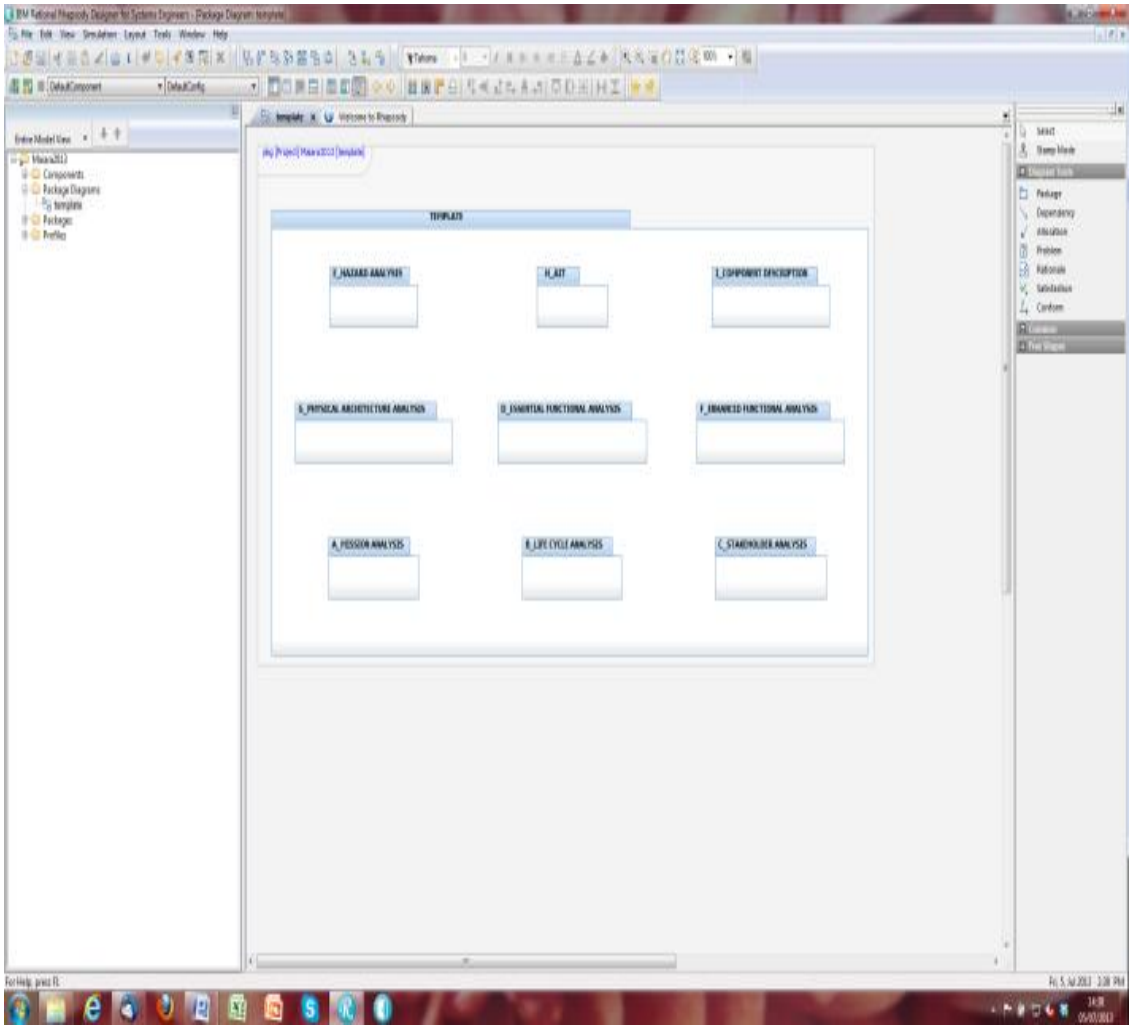


Figura 34- Estrutura do *template*.

Fonte: Flausino, 2013.

Desta forma, é possível compreender e visualizar o todo, possibilitando uma modelagem consistente. Este *template* tem como objetivo possibilitar a visão de todas as etapas, bem como a sequência lógica de sua aplicação, lembrando, que como em qualquer processo de Engenharia de Sistemas, existem interações e iterações a serem feitas.

Para modelá-lo foi necessário levar em consideração os níveis de hierarquia de um sistema, o qual contribui para o entendimento do processo e futuramente para a sua gestão também.

Este *template* desenvolvido apresenta pelo menos um diagrama para cada tipo do diagrama da linguagem de SysML, ilustrando como esses diagramas podem ser aplicados usando a abordagem desenvolvida por Loureiro (2010).

É fundamental recordar que este *template* serve para organizar o método e não para ensinar como se aplica o método, para entender a estruturação utilizada o usuário precisará ter conhecimento prévio sobre Engenharia Simultânea de Sistemas.

A seguir serão apresentadas as etapas do método proposto por Loureiro (2010), para ilustrá-las, expressando a sua hierarquia e suas relações, serão usadas algumas telas do *template*, construído durante o desenvolvimento deste trabalho.

10.1.1. Análise da missão

A missão é a declaração do objetivo a ser realizado pelo sistema, isso inclui a razão de ser do sistema. A missão usualmente é capturada pela declaração: O sistema deve [ ], sendo que entre os colchetes é descrito o propósito do sistema.

Para garantir que uma missão seja bem sucedida é preciso conhecer as necessidades dos *stakeholders*, em seguida realizar a análise da missão e ter como resultado os requisitos de missão.

Desta forma, como os requisitos de missão são muitos, algumas dezenas, deve-se usar o diagrama de requisito da linguagem SysML, para mostrar as relações entre eles, como a hierarquia, demonstrando qual é o pai e qual é seu filho, isto é, as derivações existentes no projeto.

Com a ajuda de um *software*, que de suporte a linguagem SysML, como, o *IBM Rational Rhapsody*, é possível tabelar esses requisitos automaticamente, além de poder exportar e importar para uma ferramenta de gestão de requisitos, como o *IBM Rational DOORS*.

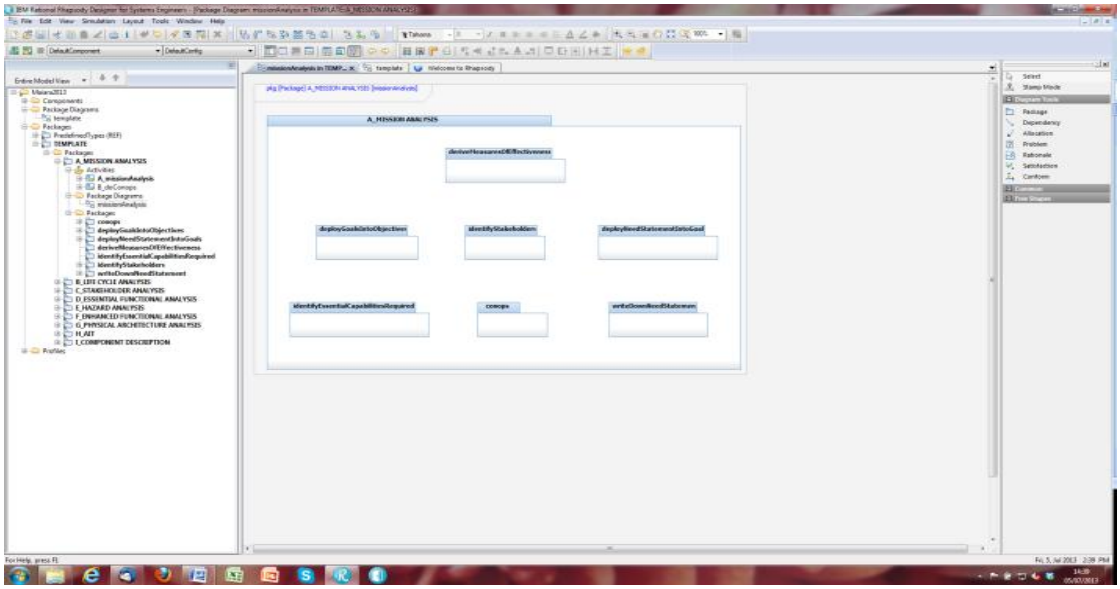


Figura 35 – Diagrama de pacote Análise da missão da linguagem SysML.

Fonte: Flausino, 2013.

10.1.2. Análise do ciclo de vida

O ciclo de vida descreve os cenários de vida de um sistema, podendo ser modelado através de um diagrama de atividade da linguagem SysML, pois descreve um comportamento do sistema.

A Figura 36 ilustra os pacotes que contem a analise do ciclo de vida de um produto.

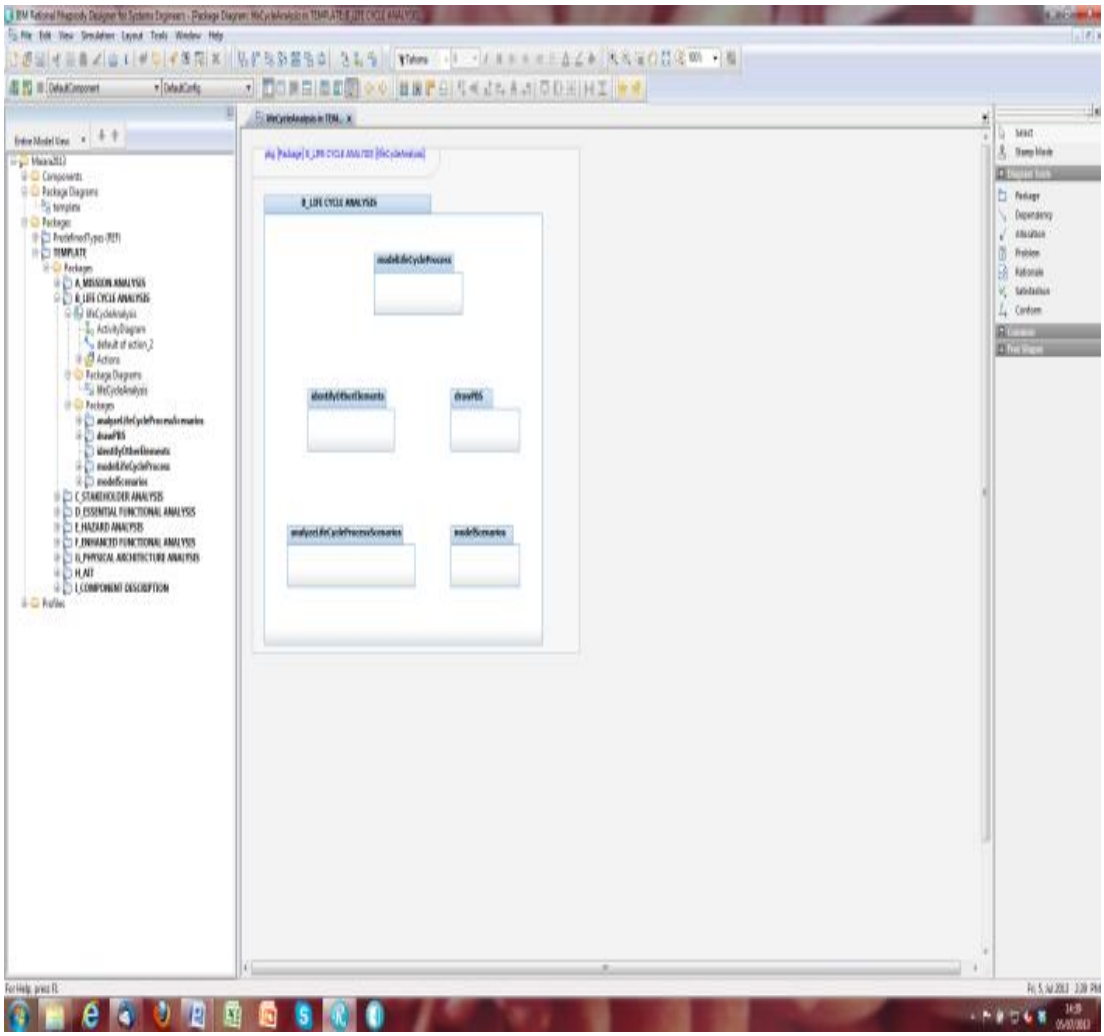


Figura 36 - Diagrama de pacote Análise do ciclo de vida da linguagem SysML (1).
Fonte: Flausino, 2013.

A seguir, será apresentado o ciclo de vida do subsistema estação terrena, do sistema CANSAT, Figura 37, o qual foi modelado em um diagrama de atividade da linguagem SysML, pois denota a dinâmica do sistema, por exemplo, a operação do sistema.

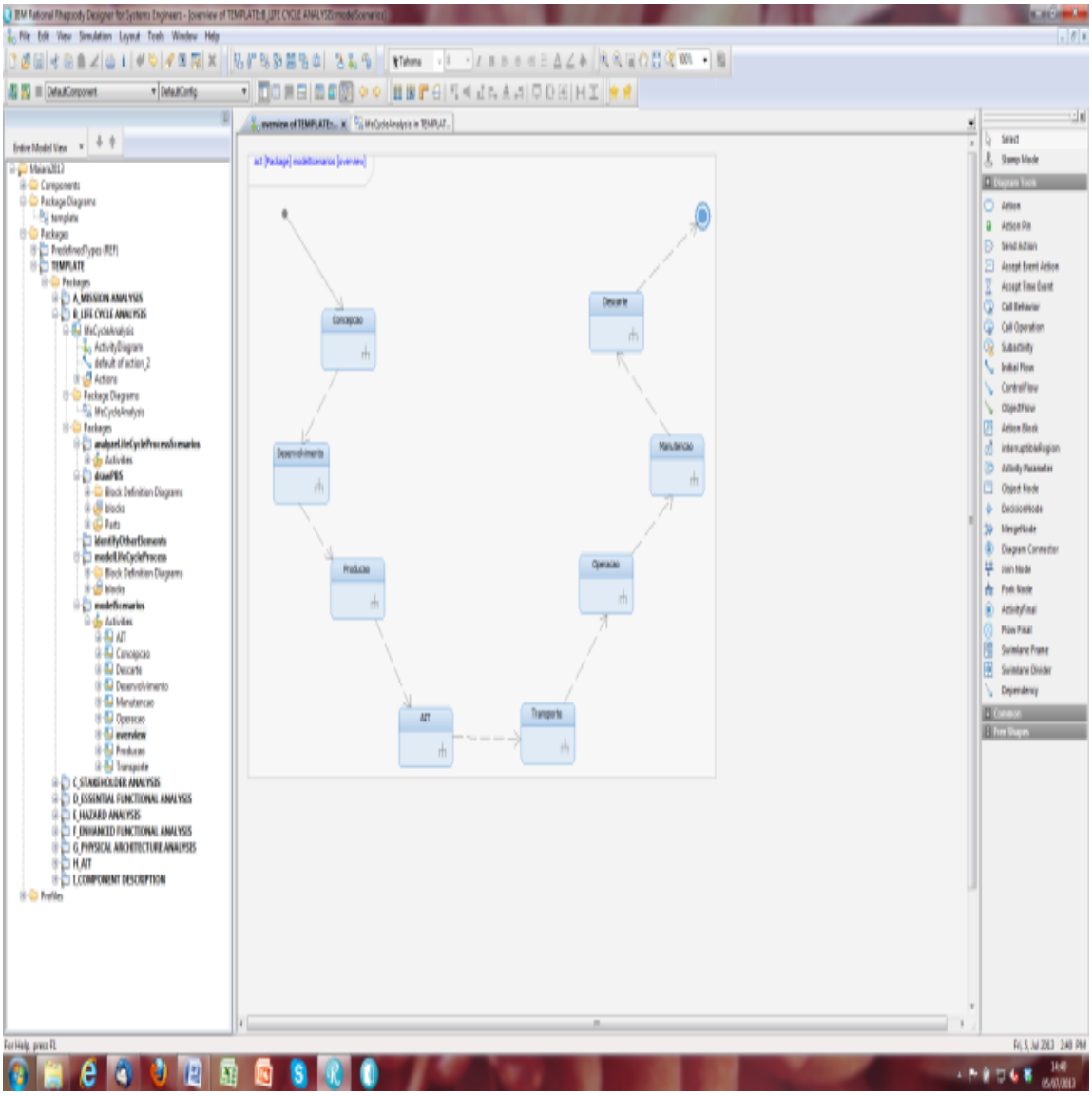


Figura 37 - Diagrama de pacote Análise do ciclo de vida da linguagem SysML (2).

Fonte: Flausino, 2013.

Os cenários, decomposição de cada processo, por sua vez, podem ser obtidos a partir do desdobramento do diagrama de atividade, Figura 37, que também podem ser modelados usando o diagrama de atividade, pois os cenários são as alternativas em cada processo do ciclo de vida, isto é, uma sequência específica de ações que ilustram o comportamento do sistema.

A modelagem dos cenários é importante, porque é através dessa modelagem que serão derivados alguns requisitos de sistema. Os processos do ciclo de vida descrevem as etapas de vida do sistema, e pode incluir outras séries de estágios, como, a fabricação do sistema, Figura 38.

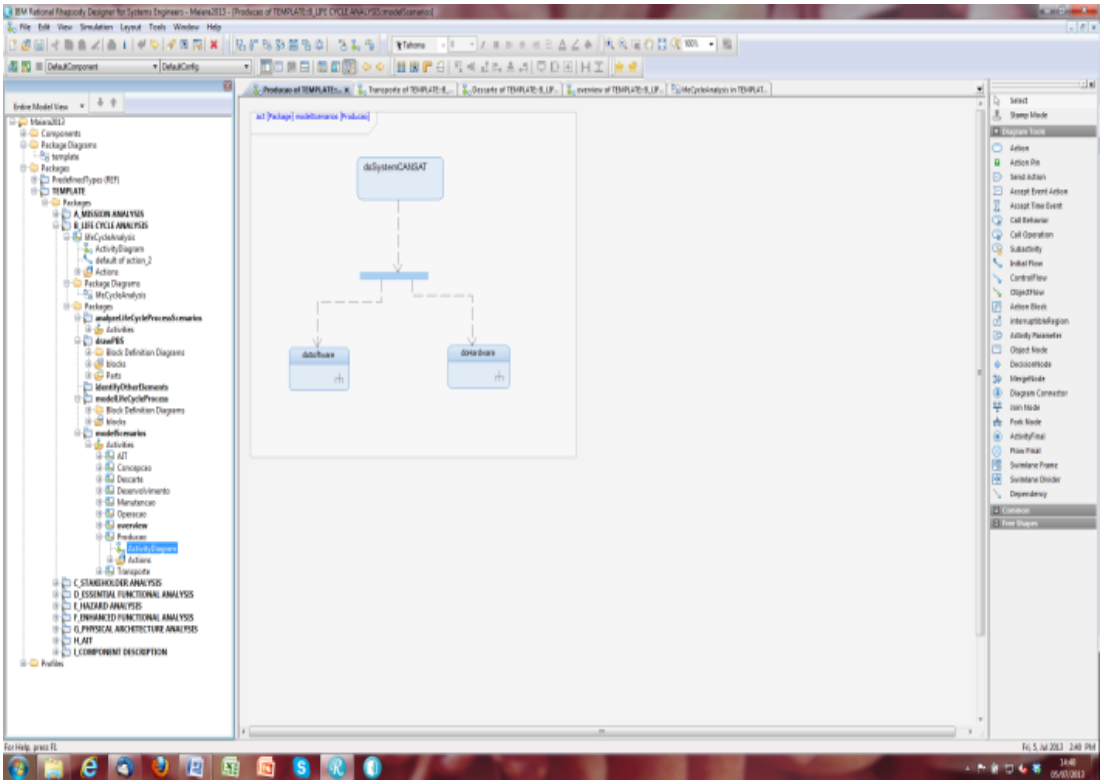


Figura 38- Diagrama de pacote Análise do ciclo de vida da linguagem SysML (3).
Fonte: Flausino, 2013.

Já o contexto funcional define o escopo em que se encontra o sistema de interesse, engloba o que está no ambiente do sistema, que por sua vez, contempla também os elementos que estão fora do sistema, mas que trocam material, energia e informação com o sistema de interesse.

O contexto funcional é identificado em cada cenário do ciclo de vida de produto, nessa etapa deve ser aplicado o diagrama de contexto, apresentado anteriormente neste trabalho.

Os elementos do ambiente podem ter diferentes estados, e o conjunto desses estados são chamados de circunstâncias. Os estados são sensoriais, por exemplo, som, imagem, posição, temperatura, isto é, as características físicas externas do sistema.

Um sistema ainda pode ter diferentes modos, dependendo das circunstâncias. Os modos são um conjunto de funcionalidades que um produto realiza dependendo das circunstâncias.

10.1.3. Análise de *stakeholders*

Os *stakeholders* são as pessoas que afetam, que são afetadas ou que têm interesse pelo produto complexo ou as organizações que participarão do ciclo de vida do produto. Tratam-se de pessoas essenciais ao planejamento do projeto do sistema.

As seguintes perguntas são feitas para identificar os *stakeholders*:

a) Quem são as pessoas, que provêem entrada para o produto em um dado cenário ou processo?
b) Quem são as pessoas, que recebem saídas do produto em um dado cenário ou processo?
c) Quem são as pessoas, que provêem controles para o produto em um dado cenário ou processo?
d) Quem são as pessoas, que provêem mecanismos para o produto em um dado cenário ou processo?

Os interesses são as expressões das necessidades dos *stakeholders*, em cada cenário dos processos do ciclo de vida do produto, por exemplo, para o motorista de um carro são interesses a segurança, o desempenho e a economia do carro.

Os *stakeholders* expressam suas necessidades e a partir desses interesses são identificados os requisitos de *stakeholders* e as *Measure of Effectiveness* (MoEs), as quais, podem incluir, por exemplo, as facilidades de uso e o desempenho de manutenção de um sistema.

Para compreender melhor uma necessidade de *stakeholders*, é necessário desdobrá-la. Assim, derivam-se as medidas dos interesses expressos por eles anteriormente, através de perguntas, usando a palavra “como?”. Isso é feito sucessivamente até que se obtenha um conjunto de medidas, por exemplo, custo, prazo, economia de combustível, produtividade. Esse processo trata-se das MoEs.

Os interesses também podem ser desdobrados usando o método *Goal, Question, Metric* (SOLINGEN & BERGHOUT, 1999), ou uma simples árvore de objetivos.

As MoEs representam uma forma de avaliar a satisfação dos *stakeholders* em relação ao produto a ser desenvolvido, ou até mesmo à organização, a qual realizará os processos dentro do escopo de desenvolvimento. A atribuição de valor e a tolerância a essas medidas geram também os requisitos de *stakeholders*.

A partir da identificação dos requisitos de *stakeholders* derivarão as capacidades, isto é, o que o sistema deve fazer e derivarão as restrições, como, o desempenho e a limitação física do sistema.

A análise de requisitos de *stakeholders* inclui a negociação com os *stakeholders* daqueles requisitos.

Os requisitos de sistema são derivados a partir da análise dos requisitos de *stakeholders*, que por sua vez são derivados da necessidade dos *stakeholders* e das MoEs.

Os engenheiros de sistemas devem ser capazes de traduzir a necessidade dos *stakeholders* em requisitos de *stakeholders* e também em requisitos de sistemas.

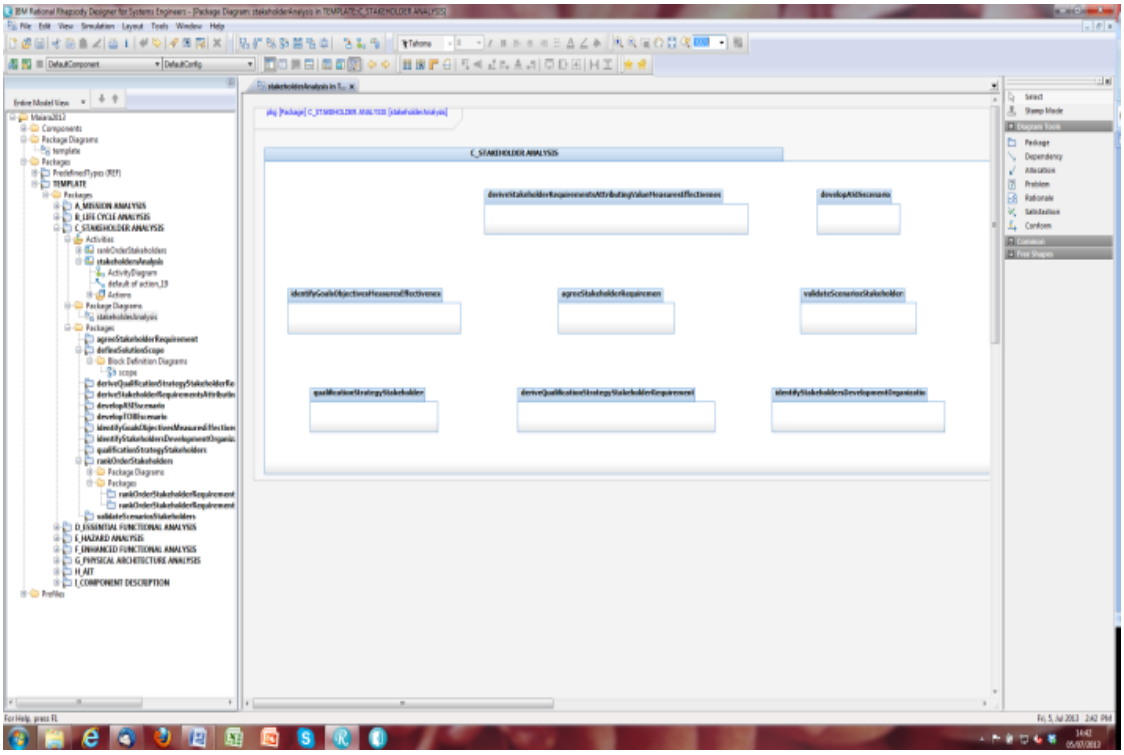


Figura 39 - Diagrama de pacote Análise de *stakeholders* da linguagem SysML.

Fonte: Flausino, 2013.

10.1.4. Análise funcional essencial

Essa etapa realiza a análise das funções do sistema para que sejam identificadas todas as sub funções necessárias para completar a função principal em estudo, isso é feito através da identificação das relações e das interfaces existentes (HALLIGAN, 2013).

É durante essa fase que é feita a definição das principais funções de um sistema, sem que se especifiquem as suas configurações físicas, as quais depois serão implementadas.

A partir do contexto funcional do produto é feita a identificação das funções internas do produto em um determinado cenário, através de uma Lista de Eventos (YOURDON, 1990), considerando os elementos do ambiente e a troca de material, energia e informações com o sistema de interesse.

A Lista de Eventos é composta pelos eventos, que são os elementos do ambiente e pelas respostas do sistema que são os resultados em relação aos eventos feitos pelos elementos do ambiente, essas respostas serão as funções essenciais do produto.

O comportamento funcional do produto indica quando e sob que condições uma determinada função é iniciada, interrompida ou encerrada, ou seja, indica a sequência no tempo da realização das funções, apresentando os aspectos causais e lógicos do sistema. É o comportamento funcional que define as transições entre estados e entre modos do sistema.

Os atributos funcionais descrevem e qualificam os elementos da estrutura e do comportamento funcional do produto e da organização. Eles podem descrever o desempenho funcional de um sistema.

O comportamento funcional permite identificar a sequência em que as funções serão executadas para os cenários dentro do escopo do esforço de desenvolvimento.

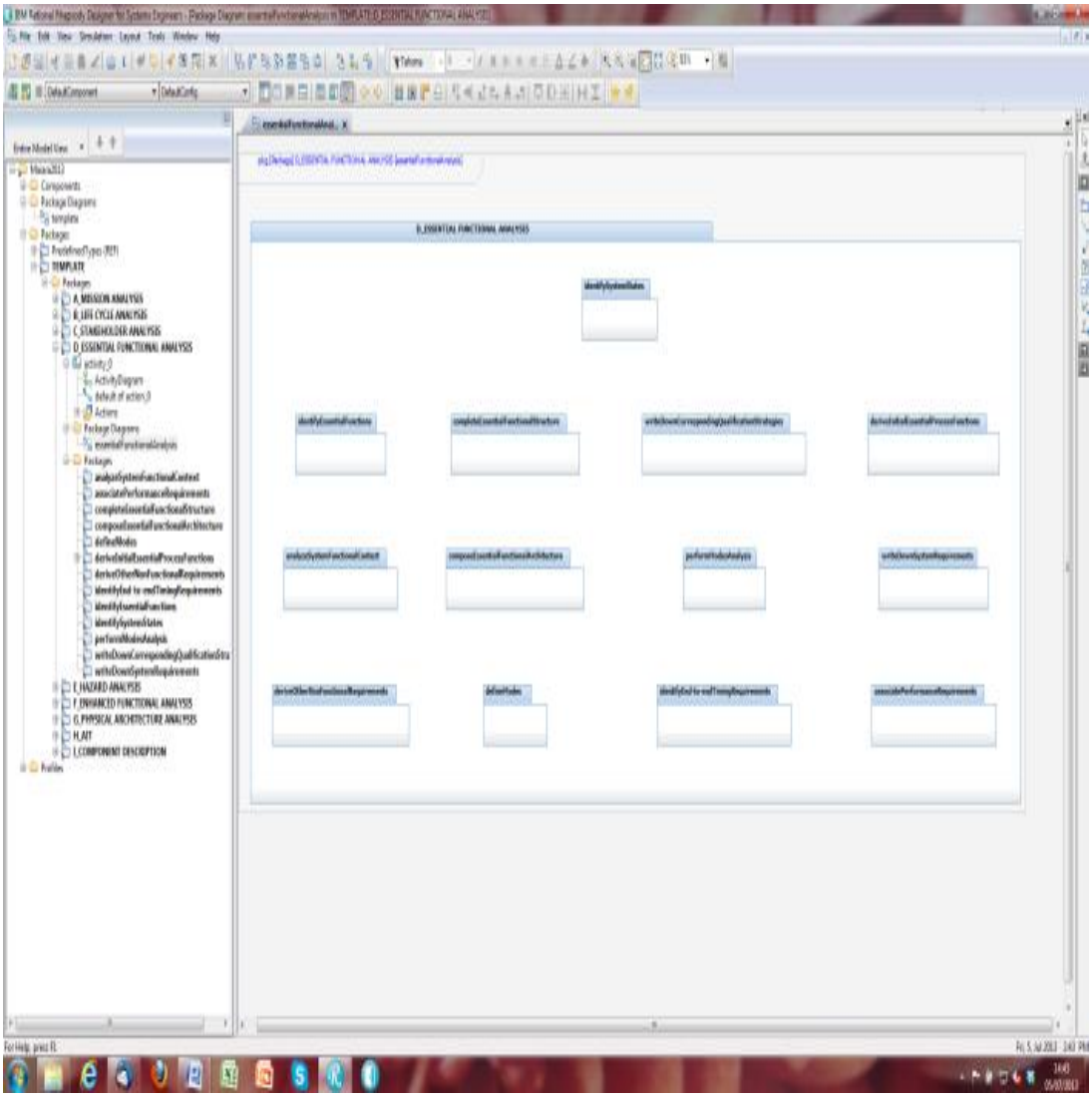


Figura 40 - Diagrama de pacote Análise funcional essencial da linguagem SysML.

Fonte: Flausino, 2013.

10.1.5. Análise de perigo

A análise de perigo é realizada para cada perigo potencial identificado a partir de circunstâncias, falhas nos fluxos entre o produto e os elementos do ambiente do sistema e falhas nas funções internas do produto.

Perigo é qualquer dano imposto pelo produto ou organização ao próprio produto ou organização, ou a qualquer elemento no ambiente do produto ou da organização. Dependendo da avaliação do perigo, funções de mitigação de risco são identificadas. Essas funções podem ser de detecção, prevenção, proteção e correção.

A tabela de falhas e de análise crítica FMECA (JURAN & GODFREY, 1998) é construída listando as circunstâncias, as falhas nos fluxos, as falhas nas funções internas, suas consequências, sua gravidade, suas causas, sua probabilidade e ocorrência, sua dificuldade de detecção e o risco computado a partir da composição das avaliações de gravidade, probabilidade e dificuldade de detecção.

Outra técnica que também pode ser aplica é o FMEA (*Failure Mode and Effect Analysis*), que tem como objetivo prevenir a ocorrência de falhas do produto ou do processo, pela analise detalhada de possíveis problemas operacionais.

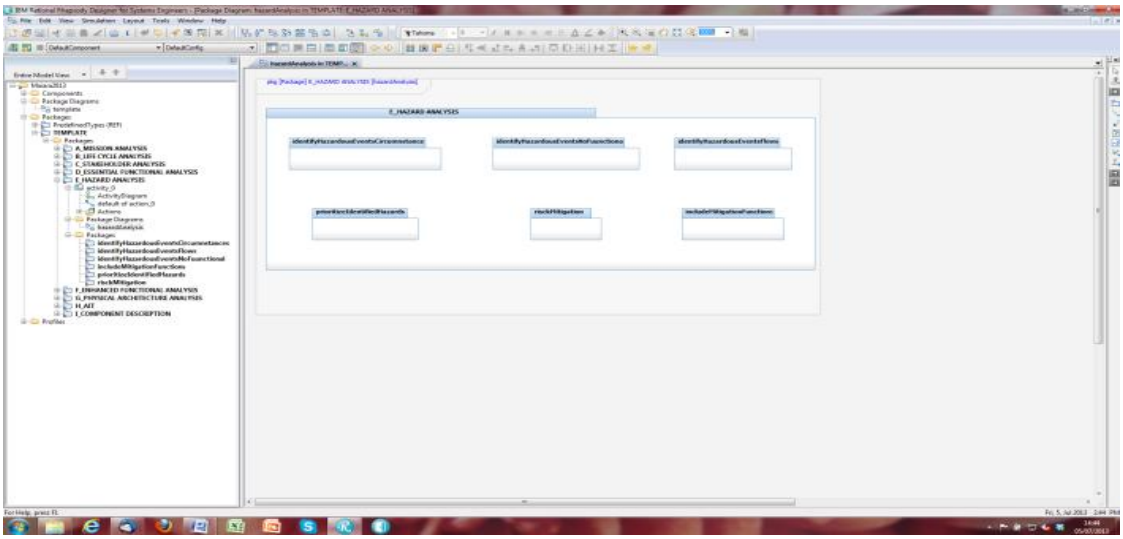


Figura 42 - Diagrama de pacote Análise de perigo da linguagem SysML.

Fonte: Flausino, 2013.

10.1.6. Análise funcional estendida

Nessa etapa o propósito da Engenharia de Sistemas é transformar os requisitos funcionais em outros requisitos, que poderão ser identificados através dessa análise funcional estendida. Mas, para isso, o engenheiro de sistemas precisa saber o que o sistema deve fazer e como deve fazê-lo. Isso é conseguido através de funções em que estejam em sequências lógicas, ou seja, em decomposição de um nível superior para um nível inferior, alocando o desempenho de funções de alto nível na inferior.

Esse processo repete-se até definir sucessivamente o menor nível funcional e os seus requisitos de desempenho, definindo assim a arquitetura em níveis crescentes de detalhes.

Desta forma, os requisitos do sistema são alocados, definido os detalhes suficientes para fornecer o projeto de verificação para apoiar o projeto integrado do sistema, ou seja, fornece rastreabilidade dos requisitos.

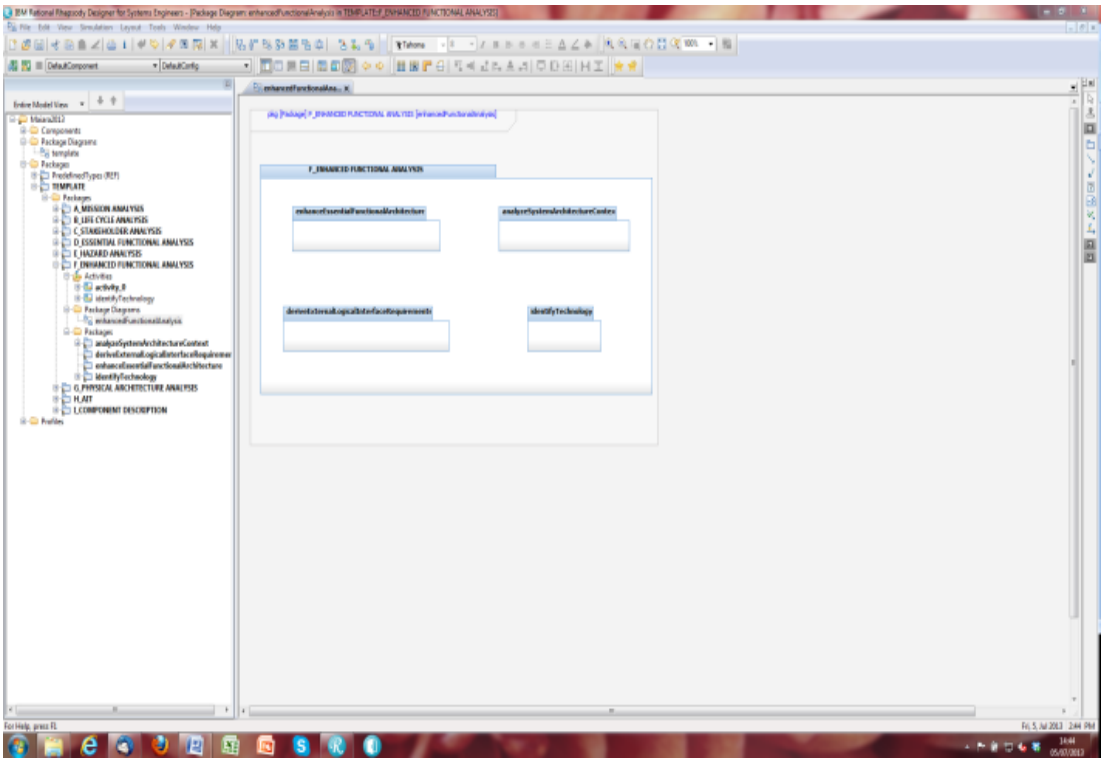


Figura 42 - Diagrama de pacote Análise funcional estendida da linguagem SysML.

Fonte: Flausino, 2013.

10.1.7. Análise de arquitetura física

A arquitetura física é o resultado do processo de Engenharia de Sistemas que apresenta os elementos do produto e da organização, sejam eles componentes, fluxos, interfaces e relacionamentos existentes, ou seja, é a implementação das funções descritas pelos elementos dos modelos no contexto funcional, como os componentes e os fluxos do sistema.

O contexto físico também é identificado em cada cenário do processo do ciclo de vida, apresentando o sistema, os elementos do ambiente, e as conexões físicas entre os componentes existentes.

A diferença entre o contexto físico e o contexto funcional, está nos elementos de ligação entre o sistema e os elementos do seu ambiente, resumindo, as interfaces. Pois, no funcional, são apresentados os fluxos de material, energia e informação. Já no físico, são apresentadas as conexões físicas entre o sistema e os elementos do seu ambiente. Portanto os fluxos do contexto funcional fluem pelas conexões do contexto físico.

As conexões físicas entre o sistema e os elementos do ambiente, definem os requisitos de interface externa de produto. A implementação provê as entradas necessárias para os processos de projeto detalhado, isto é, descrição de componentes, como, aquisição ou reuso de componentes e plano de AIT.

Os atributos de implementação do produto descrevem os elementos dos modelos de arquitetura de implementação do produto, sejam eles partes componentes, fluxos ou interconexões físicas e acrescentam características de desempenho, ou limitações físicas a cada elemento dos modelos de implementação.

O relacionamento entre atributos funcionais e de implementação representam a alocação e a rastreabilidade entre os modelos funcionais e físicos. Os relacionamentos entre atributos de produto e de organização representam os impactos mútuos entre produto e organização, e precisam ser identificados para o desenvolvimento de soluções de engenharia simultânea de sistemas.

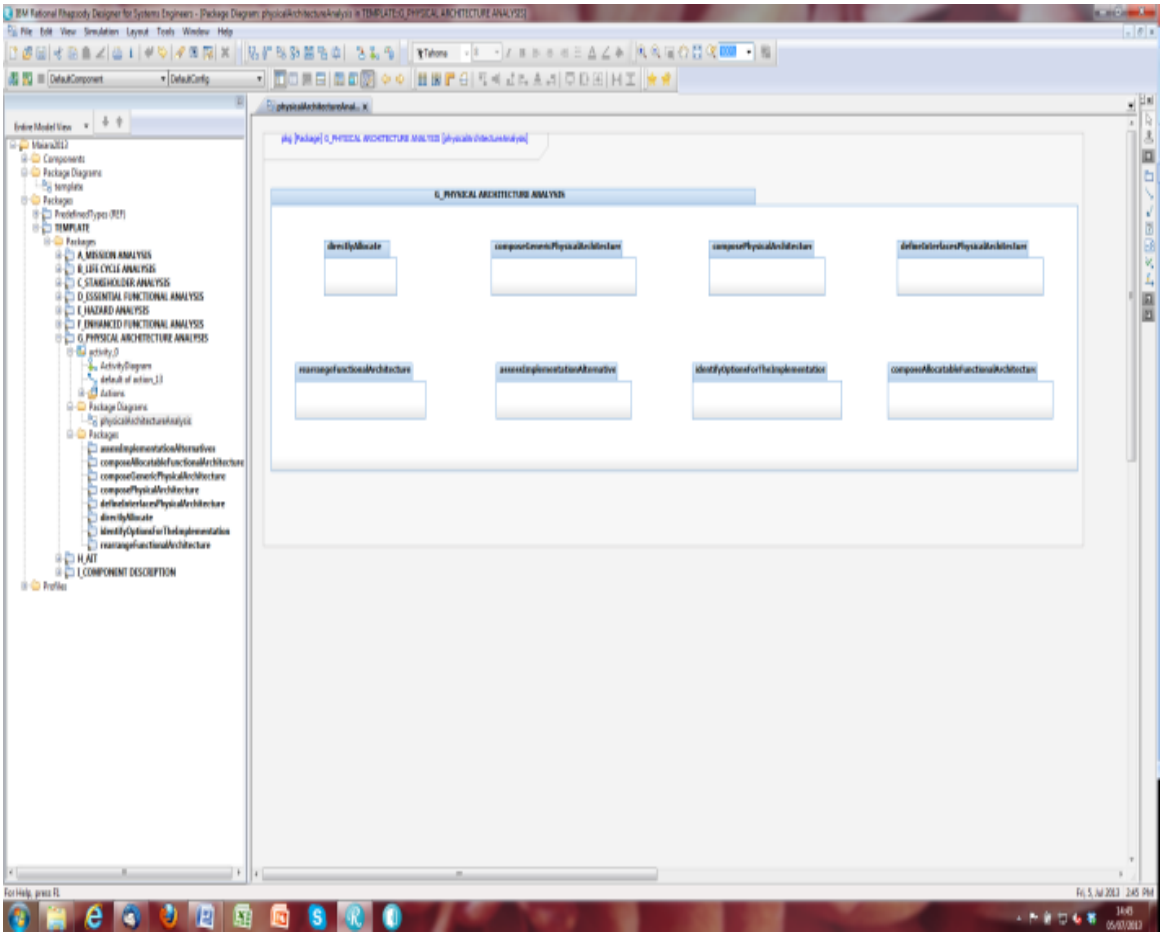


Figura 43 - Diagrama de pacote Análise de arquitetura física da linguagem SysML.

Fonte: Flausino, 2013.

10.1.8. AIT (*Assembly Integration and Test*)

O plano de AIT inclui:

a) Descrição das características gerais do produto espacial a ser desenvolvido.
b) Desenvolvimento de bancos de testes para o sistema e seus subsistemas.
c) Especificação técnica de banco de testes elétricos para o sistema e seus subsistemas.
d) Identificação das interfaces necessárias para realização da montagem, da integração e dos testes funcionais do sistema e dos seus subsistemas.
e) Projeto de equipamentos de suporte mecânico para o sistema e seus subsistemas.
f) Seleção dos parâmetros do sistema e dos subsistemas, com base na análise funcional e na arquitetura física do sistema.

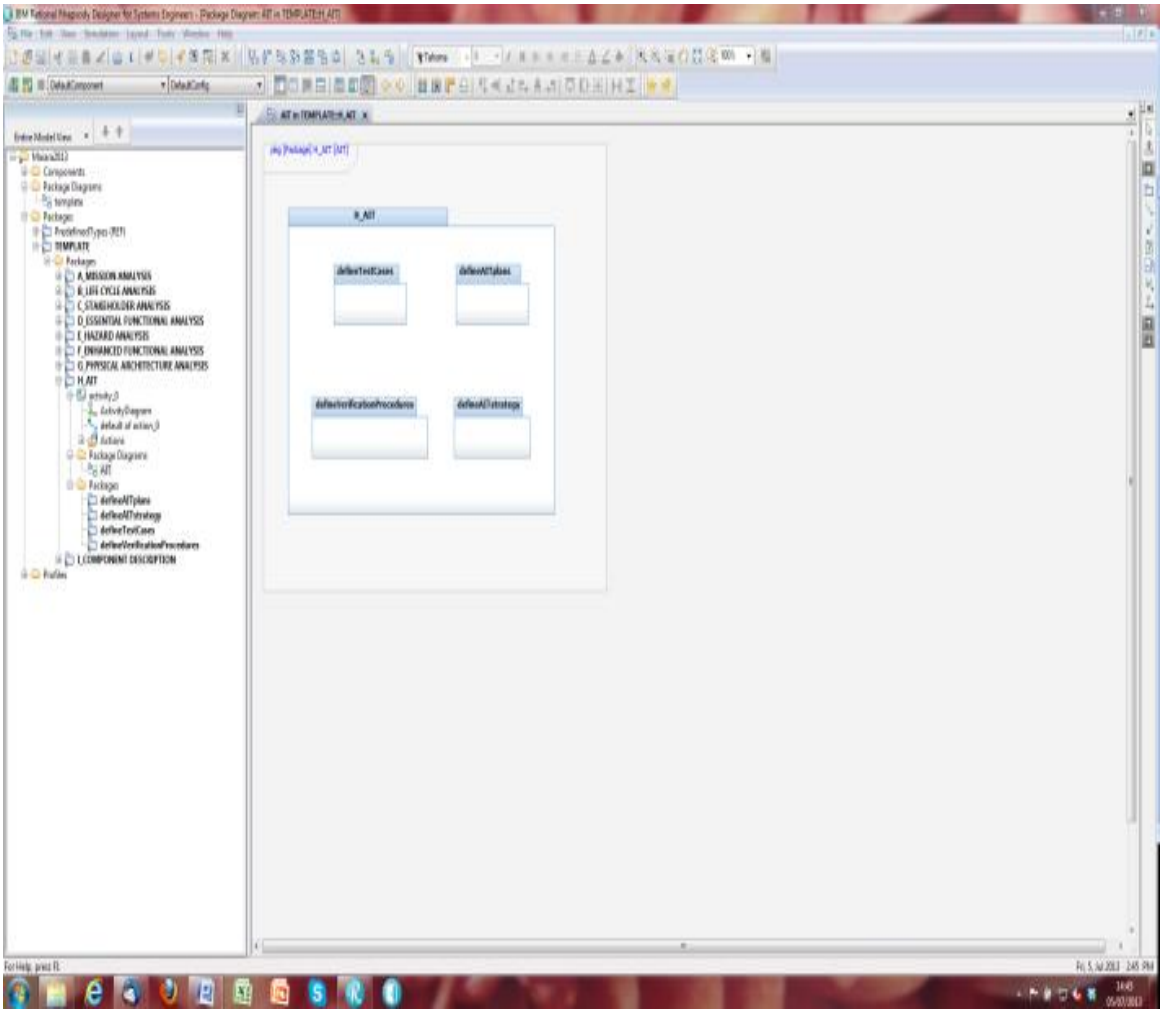


Figura 44 - Diagrama de pacote AIT da linguagem SysML.

Fonte: Flausino, 2013.

7.1. 9. Descrição de componentes

Na última fase do método de Engenharia Simultânea de Sistemas é feita a especificação de todos os componentes do sistema de interesse, pois já foi escolhida qual alternativa de arquitetura será implementada no projeto.

Esse é o nível mais detalhado do sistema, encontram-se todas as características dos componentes, bem como todas as características físicas das interfaces dos subsistemas.

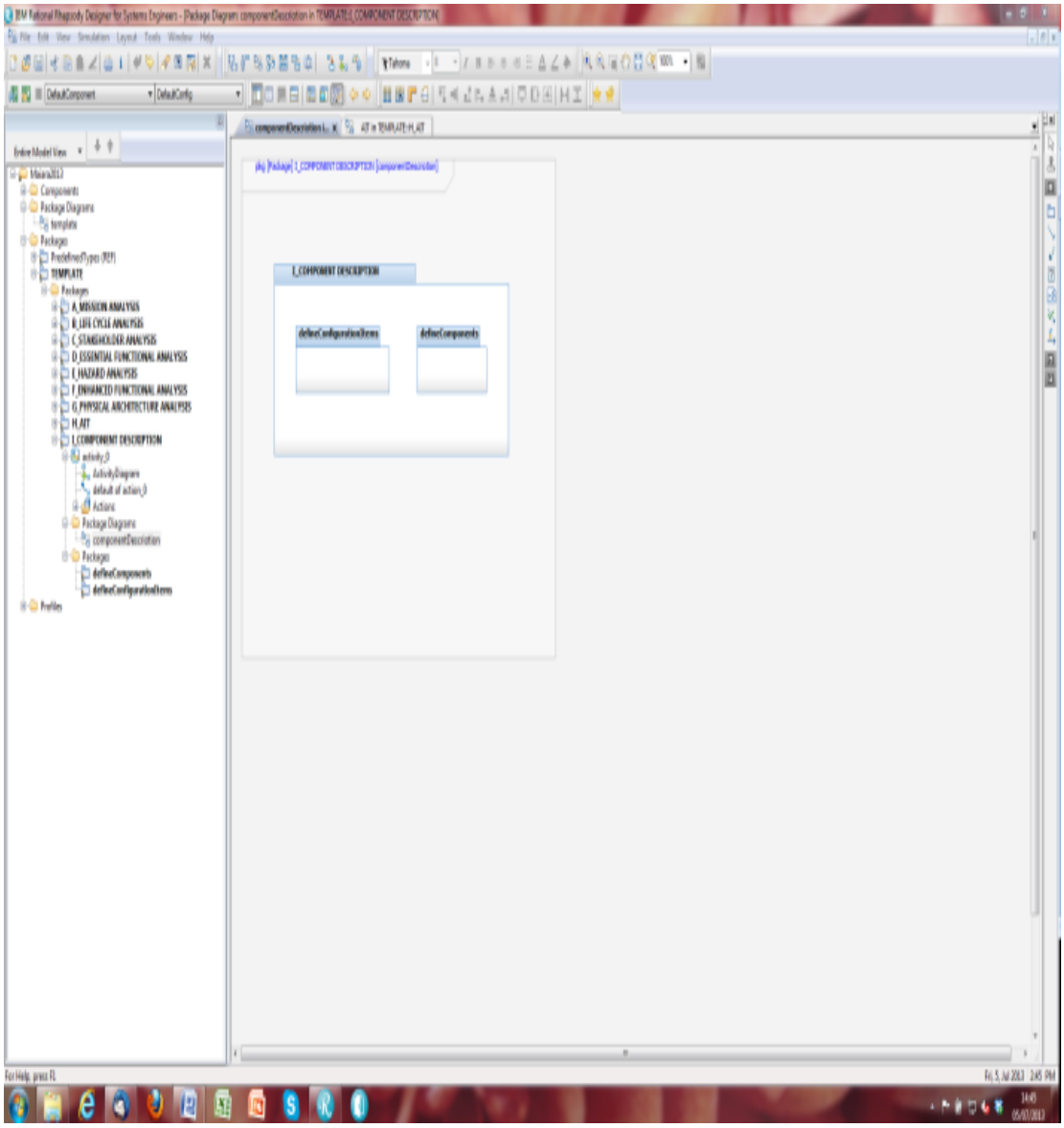


Figura 45 - Diagrama de pacote Descrição de componente da linguagem SysML.

Fonte: Flausino, 2013.

11. TRABALHO FUTURO

Como trabalho futuro espera-se que outras adaptações mais avançadas sejam feitas em relação à modelagem apresentada neste trabalho, deixando o *template* mais próximo do método aqui apresentado.

Neste trabalho, buscou se a correspondência de qual diagrama da linguagem SysML que poderia ser usado em cada etapa do método proposto por Loureiro (1999), no próximo trabalho, sugere construir novos tipos de estereótipos para que novas adaptações sejam realizadas.

Outra indicação é dar prosseguimento ao desenvolvimento da modelagem também das organizações que implementam o ciclo de vida do produto.

12. CONSIDERAÇÕES FINAIS

Este trabalho buscou realizar a modelagem, utilizando a linguagem SysML, de um produto espacial seguindo o método de Engenharia Simultânea de Sistemas de Loureiro (1999), como este foi modelado primeiramente em análise estruturada, foi preciso um estudo constante para realizar adaptações na modelagem, bem como criações de novos diagramas e relacionamentos, já que a linguagem SysML permite isso.

A linguagem SysML implementada com a abordagem da Engenharia Simultânea de Sistemas torna possível a modelagem de sistemas complexos desde a sua análise da missão até a sua descrição de componentes, ou seja, essa linguagem cobre todos os as fases da abordagem sistemática.

A Engenharia Simultânea de Sistemas, busca dar suporte a análise das etapas de desenvolvimento, garantindo uma solução balanceada entre os *stakeholders*, como também evitando os desperdícios de recursos, como, o tempo, ajudando a administrar os riscos.

Assim, a Engenharia Simultânea de Sistemas tem se tornado cada vez mais requisitada, devido à evolução das tecnologias de implementação e ao aumento da complexidade dos próprios sistemas. Logo, novos métodos têm sido desenvolvidos na tentativa de diminuir custos, riscos e tempo gastos nas etapas de modelagem e desenvolvimento do projeto.

Essa abordagem de levar em consideração o todo para analisar uma solução está cada vez mais presente tanto no mundo empresarial como no acadêmico e científico.

Os objetivos da Engenharia Simultânea de Sistemas que a linguagem SysML cobre são:

a) Conhecer e analisar as necessidades do *stakeholders*.
b) Conhecer funções, interfaces, arquiteturas do sistema.
c) Manter rastreabilidade dos requisitos.
d) Realizar *trade-off*.
e) Sintetizar as alternativas de solução do sistema

A linguagem (modelagem gráfica) é a interface entre o abstrato (significado) e o concreto (modelo), sem ela não é possível transmitir a informação nem rastreá-la posteriormente nas etapas que a Engenharia Simultânea de Sistemas atua.

Quando se encontra algum obstáculo na modelagem, o problema geralmente refere-se ao sistema complexo, porque a modelagem não é a solução para desenvolvimento do produto, mas sim uma técnica indispensável para analisá-lo, uma vez que, é através da modelagem que alguns requisitos são extraídos, por exemplo, os requisitos de interface. Com o auxilio dessa técnica também é possível a ter a rastreabilidade dos requisitos, pois o modelo prova a consistência do que foi planejado.

Para modelar na linguagem SysML, garantindo um trabalho de qualidade, é necessário:

a) Conhecer bem a linguagem.
b) Conhecer bem o sistema a ser modelado.
c) Dominar uma ferramenta computacional.

A linguagem SysML só se torna completa com a criação de estereótipos, e possui grande potencial ainda a ser desenvolvido, ou seja, essa linguagem ainda precisa de adaptações. No entanto, ela já possui as recursos suficientes para uma implementação do método, pois ela permite ao profissional que sejam criados novos tipos de diagramas, já que se trata de uma linguagem não prescritiva, ao contrário da Análise Estruturada que é prescritiva.

Como resultados esperados no início deste projeto, conclui-se que esta pesquisa conseguiu atingi-los, são eles:

a) Capacitação na linguagem SysML.
b) Capacitação em Engenharia Simultânea de Sistemas Espaciais.
c) Treinamento em ferramentas computacionais de suporte à Engenharia de Simultânea de Sistemas Espaciais.

Não pode dizer que o trabalho acaba por aqui, pois a própria linguagem SysML ainda tem muito a melhorar e contribuir para a Engenharia Simultânea de Sistemas.

A seguir são apresentados exemplos de projetos relacionados à Engenharia de Sistemas, nos quais a linguagem SysML contribuiria de forma positiva:

ENGESIS (Engenharia, Gestão e Simulação de Sistemas Espaciais)

Esse grupo foi criado em 2010, a partir da união dos esforços realizados pelos pesquisadores e Professor de grupos análogos já existentes nas instituições participantes, objetivando a realização de estudos em temas relacionados com o ciclo de vida completo (o projeto, a construção, a operação, o monitoramento, o controle, a análise e a melhoria contínua) do desenvolvimento de modelos, produtos e processos da área espacial ou sistemas semelhantes de alta tecnologia.

As principais áreas do conhecimento envolvidas nos estudos realizados pelo grupo ENGESIS são: a Engenharia da Produção, a Engenharia e Gestão de Desenvolvimento de Software, Gestão de Projetos, Gestão por Processos e Modelagem e Simulação de Sistemas. A característica principal que o distingue é a utilização de uma abordagem interdisciplinar, denominada Ciência e Tecnologia de Processos, na qual os conceitos, métodos e ferramentas dessas áreas são integrados e aplicados para a solução de problemas da área espacial.

Com a linguagem SysML é possível modelar e documentar todos os processos propostos por este grupo, o que garante uma melhor compreensão do todo, por todos os integrantes do grupo, devido a visão sistêmica que a linguagem oferece.

LSIS (Laboratório de Engenharia Simultânea de Sistemas)

Esse tem por objetivo integrar ferramentas de análise de *stakeholders*, engenharia de requisitos, modelagem conceitual de sistemas, modelagem física de sistemas, projeto CAD de produtos, ferramentas de verificação e validação de sistemas.

Essas ferramentas darão suporte a um processo de engenharia simultânea de sistemas que antecipa, para as etapas iniciais do desenvolvimento do sistema, os requisitos dos processos do ciclo de vida do sistema.

Esse processo, apoiado pelas ferramentas descritas, também tem o potencial de serem utilizadas para o fornecimento de serviços para clientes do LIT, tais como a engenharia do INPE, a AEB, empresas fornecedoras do programa espacial e de outros programas complexos, bem como pelos clientes de ensaios do LIT.

O *template* elaborado neste trabalho, na linguagem SysML, ajudará o desenvolvimento desse projeto, que também segue o método desenvolvido por Loureiro (1999). Isso se trata de um dos resultados esperados desta pesquisa.

Projeto SACI (Satélite Avançado Comunicações Interdisciplinares)

A motivação desse projeto foi à necessidade de viabilizar um sistema avançado de tecnologia educacional de alcance nacional para o Brasil.

A documentação desse projeto encontra-se na biblioteca central do INPE em São José dos Campos. São vários relatórios em três idiomas diferentes (português, inglês e espanhol), o que já indica uma falta de padrão na documentação em relação ao uso da linguagem SysML. Além disso, os documentos estão desatualizados e desgastados pelo tempo.

As informações que eles contêm são importantíssimas, pois se trata do projeto mais revolucionário em educação usando tecnologia, já idealizado no INPE pelo seu diretor da época Dr. Fernando Mendonça. É evidente que na época não existia a linguagem SysML. Mas, se esse projeto tivesse sido realizado nos dias de hoje o seu ganho no projeto seria muito maior.

Uma vez que, partir de documentos foi possível conhecer um relato sobre o Projeto SACI. Esse relato demonstra a importância que a modelagem tem no desenvolvimento de um produto complexo, nos relatórios técnicos do projeto são descritos como as decisões mudavam rapidamente, isso teve como conseqüência a perda de rastreabilidade dos requisitos. Isso ocorreu, pois na época, década de 1960, não havia uma ferramenta computacional que desse suporte a modelagem, bem como não havia uma linguagem gráfica que abrangesse todo o ciclo de vida do produto.

Desta forma, o sistema em desenvolvimento começou a ficar com *gaps* e os engenheiros não conseguiram mais aplicar a abordagem de Engenharia de Sistemas por falta de técnicas de modelagem que permitissem visualizar o todo.

Esse é apenas um exemplo, dentre muitos outros, que comprovam a importância da modelagem como um dos pilares da Engenharia de Sistemas. Por isso, também é importante ter a rastreabilidade de requisitos para que qualquer mudança seja comunicada, mostrando o que será afetado direta ou indiretamente com as atualizações feitas.

REFERÊNCIAS BIBLIOGRÁFICAS

3SL. Cradle user manual, 2010. Disponível em: < http://www.threesl.com/> Acesso em: ago. 2011.

BALMELLI. **An overview of the systems modeling language for products and systems development**. IBM Technical Report. 2006.

Cradle
Disponível em:
<http://www.threesl.com/pages/Cradle/English/Content/Products/workbench.php>
Acesso em: ago. 2012.

EA Trial
Disponível em: <http://www.sparxsystems.com/products/ea/trial.html>
Acesso em: ago. 2012.

EA SysML Disponível em:
<http://www.sparxsystems.com/products/mdg/tech/sysml/index.html>
Acesso em: ago. 2012.

EA
Disponível em: <http://www.sparxsystems.com/products/ea/index.html>
Acesso em: ago. 2012.

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **A practical guide to SysML**: the systems modeling language. The MK/OMG Press. Elsevier. Amsterdam, 2009.

FULINDI, J. **Auxílio computacional a um processo de engenharia simultânea de sistemas espaciais**. 2011. 273p. (sid.inpe.br/mtc-m19/2011/04.04.12.08-TDI). Dissertação (Mestrado em Engenharia e Gerenciamento de Sistemas Espaciais) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2011. Disponível em: <http://urlib.net/8JMKD3MGP7W/39EQGDP>. Acesso em: ago. 2011.

HALL, Arthur D. **A Methodology for systems engineering**. Van Nostrand. New York, 1962.

HOLT, J.; PERRY, S. **SysML for systems engineering**. IET. Stevenage. 2008.

IBM. DOORS user manual, 2010.
Disponível em: < http://www-01.ibm.com/software/awdtools/doors/>
Acesso em: ago. 2011.

IBM. Rhapsody 7.0 user manual, 2010.
Disponível em: < http://www-01.ibm.com/software/awdtools/rhapsody/>
Acesso em: ago. 2011.

IBM Trial DOORS
Disponível em:
<https://www.ibm.com/developerworks/community/blogs/requirementsmanagement/entry/doors_and_testing_your_options?lang=en>
Acesso em: ago. 2012.

IBM Trial Rhapsody Disponível em:
<http://www.ibm.com/developerworks/downloads/r/rhapsodydeveloper/?S_CMP=rnav&ca=qapromo&s0rat-b0rat-l0rat-d0rat-n021-o0F129911Y76724U66-g0usen>
Acesso em: ago. 2012.

INCOSE San Francisco Bay Area Chapter. **Systems engineering handbook**: a how to guide for all systems engineers. San Francisco, 1998.

JURAN, J.M.; GODFREY, A.B. **Jura's quality handbook**. MCGRAW- HILL. United States of America, 1998.

LANE, J. A.; BOHN T. Using sysml modeling to understand and evolve systems of system. **Systems Engineering**, Hoboken, v. 16, n.1, p. 87-98, spring 2013.

LOUREIRO, G. Engenharia de sistemas. 2011. São José dos Campos. ITA. Notas de aula do curso de graduação de Engenharia Aeroespacial. ITA, 2011.

LOUREIRO, G. Engenharia de sistemas. 2011. São José dos Campos. ITA. Notas de aula do curso de pós-graduação de Engenharia de Sistemas. ITA, 2011.

LOUREIRO, G. Engenharia de sistemas. 2012. São José dos Campos. ITA. Notas de aula do curso de pós-graduação de Modelagem de Sistemas. ITA, 2012.

LOUREIRO, G. **A systems engineering and concurrent engineering framework for the integrated development of complex products**. 1991. (PhD Thesis) - Loughborough University, Loughborough, UK. 1999.

LOUREIRO, G. 2010. Lessons learned in 12 years of space systems concurrent engineering. In: INTERNATIONAL ASTRONAUTICAL CONGRESS, 61. 2010. **Proceedings ....** Prague, CZ. 2010.

Magic Draw Disponível em: <https://www.magicdraw.com/main.php?ts=navig&cmd_show_newandnoteworthy=1&version=12.0&product=magicdraw&menu=new_and_noteworthy>
Acesso em: ago. 2012.

MEDINA, M.; FERTIG, C. **Algoritmos e programação**: teoria e prática. Novatec. São Paulo, 2005.

Modelio Disponível em: <http://www.modeliosoft.com/en/download/system-architect-solution.html>
Acesso em: ago. 2012.

Modelio Trial Disponível em: <http://www.modeliosoft.com/en/download/free-downloads.html>
Acesso em: ago. 2012.

MODTELLER M.; AUSTIN M.; GHODSSI R.; YANG S. Plataform for Engineering Experiemental biomedical systems, **INSIGHT**, Hoboken, v. 15, n.4, p. 9-15, dec 2012.

OMG SYSML, 2012. Disponível em: < http://www.omgsysml.org/> Acesso em: jul. 2012.

PAIM, R.; CARDOSO, V.; CAULLIRAUX, H. M.; CLEMENTE, R., Gestão de Processos. Porto Alegre: Bookman, 2009.

Papyrus SysML Disponível em: <http://www.papyrusuml.org/scripts/home/publigen/content/templates/show.asp?P=128&L=EN>
Acesso em: ago. 2012.

PIHERA L. D.; ENDER T.; Taylor B. e al. Applying systems Engineering to improve extracorporeal membrane oxygenation therapy, **INSIGHT**, Hoboken, v. 15, n.4, p. 22-29, dec. 2012.

PRASAD, B. **Concurrent engineering fundamentals**: integrated product and process organization. PTR Prentice Hall. New Jersey, 1996

SAGE, Andrew P. **Systems Engineering**. John Wiley & Sons, Inc. New York, 1992.

WEILKIENS, Tim. **Systems engineering with SysML/UML**: modeling, analysis, design. The MK/OMG Press. Elsevier. Amsterdam, 2009.

ANEXO A - Glossário

Algoritmo: Sequência de passos que visam atingir um objetivo bem definido, ou seja, já que foi estruturado.

Atividades: Comportamento expresso como um conjunto de ações conectadas/relacionadas por fluxos de controle.

Capability: Requisito de como o sistema deve ser capaz de realizar determinada ação.

Caso de uso: Discrição de um conjunto de ações, incluindo variantes, que um sistema realiza, fornecendo o resultado observável do valor de um ator.

Cenário: Sequência específica de ações que ilustram o comportamento de um sistema.

Ciclo de vida: Estrutura conceitual, ou seja, número de estágios/fases de um sistema.

Componente: Parte substituível de um sistema com o qual está em conformidade e proporciona a realização de um conjunto de interfaces.

Contexto: Conjunto de elementos relacionados para um determinado propósito, como especificar uma operação; Visão particular de um *stakeholder*.

Diagrama: Apresentação gráfica de um conjunto de elementos, em geral representada como um gráfico conectado de vértices (itens) e arcos (relacionamentos).

Diagrama de atividade: Diagrama que mostra fluxos de controle e de dados entre atividades, isto é, abrange a visão dinâmica do sistema.

Diagrama de caso de uso: Diagrama que mostra um conjunto de casos de uso, atores e seus relacionamentos.

Diagrama de bloco: Diagrama que mostra um conjunto de blocos e seus relacionamentos, isto é, abrange a visão estática de um sistema.

Diagrama de máquina de estado: comportamento que especifica a sequência de estados pelas quais um sistema passa durante seu tempo de vida como resposta a eventos.

Diagrama de sequência: Diagrama de interação que dá ênfase a ordenação temporal de mensagens trocadas em um sistema.

Domínio: Área de conhecimento, caracterizada por um conjunto de conceitos e terminologia compreendidos pelos participantes dessa área.

Engenharia de Sistemas: Disciplina multidisciplinar que visa diminuir os impactos negativos na fabricação de um sistema complexo desde a sua concepção.

Engenharia Simultânea: Abordagem sistemática para o desenvolvimento integrado e entre o projeto e o seu produto.

Escopo: Matéria/ambiente a ser estudado, pode ser entendido como o contexto do sistema.

Estado: Condição/situação ao longo da vida de um produto, realização de uma atividade ou algum evento.

Estereótipos: Extensão do vocabulário das linguagens UML e SysML, que permite a criação de novos tipos de diagramas e relacionamentos.

Evento: Especificação de uma ocorrência significativa, que tem localização no tempo e no espaço; no contexto de uma máquina de estados, a ocorrência de um evento é capaz de ativar a transição de um estado para outro.

Framework: Padrão de arquitetura que fornece um *template* extensível para aplicações em um domínio determinado.

Modelagem: Algoritmo que demonstra /expõe as características de um sistema.

Modelo: Simplificação da realidade, criado com a finalidade de proporcionar uma melhor compreensão do ambiente que está sendo estudado; Abstração semanticamente próxima de um sistema.

Operação: Implementação de um serviço que pode ser solicitado por um subsistema, com a finalidade de afetar o comportamento do sistema.

Pacote: Mecanismo de propósito geral para a organização de elementos de um sistema.

Parâmetro: Especificação de uma ou mais variáveis que pode(m) ser(em) alterada(s).

Processo: Fluxo de controle, que pode ser executado concorrentemente com outros processos; Atividades inter-relacionadas.

Requisito: Declaração semântica de uma necessidade, permitindo acrescentar novas regras ou modificar as existentes.

Sistema: Conjunto de elementos organizados para a realização de um propósito específico, que pode der descrito por modelos, provavelmente, sob diferentes pontos de vista, ajudando a compreender o todo.

Stakeholders: Pessoas interessadas em resolver o problema, que são essenciais ao planejamento do projeto.

Subsistema: Agrupamento de elementos, que constituem uma especificação do comportamento oferecido por outros elementos contidos nesse agrupamento.

Template: Modelo ou elemento parametrizado.

Visão de projeto: Visão da arquitetura de um sistema, abrangendo os subsistemas, e interfaces, que formam o domínio do problema e sua solução, a visão de projeto também abrange os requisitos funcionais do sistema.

ANEXO B - Tabela descritiva do método de Loureiro

Activities	Method
Identify the system of interest	Architecture block diagram
Review similar developments	Research
Identify the deliverable products	Architecture block diagram and a list
Mature the technical baseline	Baseline sequence plan
Determine the technical integration approach	N2
Plan hardware and software integration	ABD
Develop the work breakdown structure	WBS
Organize the technical team	Organization charts
Develop the responsibility assignment matrix	Responsibility assignment matrix
Schedule the technical work	Gantt chart
Define work packages	Activity grouping and analysis
Develop the technical resource estimate	Gantt chart
Define the discrete technical plans	Planning
Prepare the systems engineering management plan	SEMP
Prepare the software management plan	SMP
Prepare the safety and mission assurance plan	S&MAP
Get stakeholder commitments to technical plans	Review meeting
Issue autorized technical work directives	Comm means
Report technical status	Reporting
Document and iterate	Minutes
Write down the need statement	Writing up
Identify, in the need statement, the essential capabilities required	Inspection
Identify stakeholders for these essential capabilities	IDEF0
Rank order stakeholders	AHP
Deploy the need statement into goals	Objective tree, stakeholder interviews
Deploy the goals into objectives	Objective tree (GQM), stakeholder interviews
Derive measures of effectiveness	Objetive tree (GQM)
Derive stakeholder requirements by attributing value to the measures of effectiveness	Writing up
Derive qualification strategy for the stakeholder requirements	How do I assess...?
Agree stakeholder requirements and qualification strategy with stakeholders	Stakeholder interviews
Rank order stakeholder requirements for each stakeholder	AHP
Rank order stakeholder requirements by composing stakeholder and requirements ranking	Mathematics

	Define solution scope	Context analysis
	Develop 'as is' scenario from the mission statement and goals	Scenario analysis
	Develop 'to be' scenarios from mission statement and goals	Scenario analysis
	Validate scenarios with stakeholders	Stakeholders interviews
	Identify stakeholders of the development organization (higher level systems, project management, regulatory bodies, customer, user, technology developers)	IDEF0
	Rank order stakeholders	AHP
	Identify goals, objectives, measures of effectiveness	Objetctive tree (GQM)
	Derive stakeholder requirements by attributing value to the measures of effectiveness	Writing up
	Derive qualification strategy for the stakeholder requirements	How do I assess...?
	Agree stakeholder requirements and qualification strategy with stakeholders	Stakeholder interviews
	Rank order stakeholder requirements for each stakeholder	AHP
	Rank order stakeholder requirements by composing stakeholder and requirements ranking	Mathematics
	Synthesize system operational architecture	Morphological chart
	Draw system operational architecture trade options	Trade tree
	Represent system operational architecture structure and interfaces	Block diagram, N2
	Trace system operational architectural elements to scenario capabilities	Traceability table
	Assess system operational architecture options	Decision analysis
	Choose a few system operational architectures	Decicion analysis
	Analyze life cycle process scenarios for the system of interest in the system operational architecture	Scenario analysis, behaviour diagram
	Model life cycle process and its scenarios	IDEF0
	Identify other elements that will need to be developed so that the system can accomplish its life	IDEF0
	Draw a product breakdown structure tree with the system and other expected deliverables	PBS
	Identify stakeholders for each life cycle process scenario as the sources of mechanisms, controls, inputs and outputs	IDEF0
	Rank order the stakeholders	AHP
	Identify goals, objectives and measures of effectiveness for each stakeholder	Objective tree, GQM
	Derive stakeholder requirements by attributing value to the measures of effectiveness	Wrting up
	Derive qualification strategy for the stakeholder requirements	How do I assess...?
	Agree stakeholder requirements and qualification strategy with stakeholders	Stakeholder interviews
	Rank order stakeholder requirements for each stakeholder	AHP
	Rank order stakeholder requirements by composing stakeholder and requirements ranking	Mathematics
	Analyze all stakeholder requirements identifying capabilities and constraints	Requirements analysis

Define modes from sets of circumnstances	Circumnstances analysis	
Perform modes analysis	State transition diagram	
For each mode identified, identify end-to-end timing requirements	Timing tables and graphs	
For each mode identified, derive the initial essential process functions	Context analysis and list of events	
Identify essential functions from identified capabilities expressed by stakeholders	Requirements analysis	
Complete the essential functional structure	DFD	
Identify system states and describe system behaviour	STD	
Compose the essential functional architecture	DFD, STD	
Associate performance requirements to each of the identified functions	Requirements analysis from stakeholder constraints	
Derive other non functional requirements from identified constraints	Requirements analysis from stakeholder constraints	
Write down the system requirements and corresponding qualification strategies	Writing up, verification matrix	
For each mode, from the context diagram, identify hazardous events related to the circumnstances of the elements in the environment	Context analysis	
For each mode, from the context diagram, identify hazardous events related to the flows	Context analysis	
For each mode, from functional analysis, identify hazardous events related to non or malfunctioning of the identified essential functions of the system	Functional analysis	
Prioritize the identified hazards	FMECA	
For the highest priority hazards, identify mitigation protective, preventive, corrective and detection functions	FMECA	
Include mitigation functions in the essential functional architecture derived previously	Functional analysis	
Analyze system architecture context in each of the scenarios identified	Architecture context diagram	
Identify technology non specific input processing functions	DFD	
Identify technology non specific output processing functions	DFD	
Identify technology non specific human interface functions	DFD	
Identify technology non specific enabling product functions	DFD	
Enhance the essential functional architecture	DFD	
Derive external logical interface requirements	Writing up	
Compose a generic physical architecture	Architecture flow diagram, Architecture interconnect diagram	
Rearrange functional architecture so that functions or group of fucntions can be allocated to elements in the physical architecture	DFD, DSM	
Compose an allocatable functional architecture	DFD	
Directly allocate, partition or decompose functional and non functional requirements to each of the physical architecture elements	Requirements analysis, margins, budgeting	
Identify options for the implementation of elements in the generic physical architecture	Morphological chart	
Assess implementation alternatives for each element in the generic physical architecture	Decision analysis	
Compose a physical architecture	Architecture flow diagram, architecture interconnect diagram	
Derive initial lower level requirements for each element in the physical architecture	Requirements analysis and writing up	
Define interfaces in the physical architecture	N2 and ICDs	

	Define assembly, integration and verification strategy	AIT planning
	Define assembly, integration and verification plans	AIT planning
	From requirements, define test cases	AIT planning
	From test cases define verification procedures	AIT planning
	Compose an architecture block diagram	ABD
	Define components to be reused, to be COTS, to be developed internally, to be developed externally	ABD
	Define the configuration items out of the architecture block diagram	ABD
	For COTS and reused identify further qualification required and non functional requirements	Requirements analysis
	For developed components, define their requirements following the steps above	Requirements template

ANEXO C - Tabela descritiva do método de Loureiro modificada por Flausino

	Activities	Output	Method	SysML
1	MISSION ANALYSIS			
1.1	Write down the need statement	Need Statement	**Elicitation,** Writing up	X
1.2	Identify, in the need statement, the essential capabilities required	Capabilites	Inspection	X
1.3	Identify stakeholders for these essential capabilities	Stakeholder List	IDEF0	Context Diagram
1.4	Deploy the need statement into goals	Mission Goals	Objective tree, stakeholder **elicitation**	Block Diagram
1.5	Deploy the goals into objectives	Mission Objectives	Objective tree (GQM), stakeholder **elicitation**	Block Diagram
1.6	Derive measures of effectiveness	System MoEs	Objetive tree (GQM)	Block Diagram
1.7	STAKEHOLDER ANALYSIS			
CONOPS	Synthesize system operational architecture **alternatives**	Operational Architecture Alternatives	Morphological chart	
	Draw system operational architecture trade options	Architecture Trade Options ?	Trade tree	
	Represent system operational architecture structure and interfaces	Op. Arch. Alternatives Block and N2 Diagrams	Block diagram, N2	
	Trace system operational architectural elements to scenario capabilities	Capabilities traced to arch. Elements	Traceability table	
	Assess system operational architecture options	Op. Arch. Trade-off report	**Effectiveness Analysis, Risk Analysis, Trade-off analysis**	
	Choose a few system operational architectures	Selected Op. Architectures	Decicion analysis	X
2	LIFE CYCLE ANALYSIS			
2.1	Identify other elements that will need to be developed so that the system can accomplish its life	Enabling Products identification		
2.2	Model life cycle process and its scenarios	Life Cycle Model	IDEF0	Activity Diagram
2.3	Analyze life cycle process scenarios for the system of interest in the system operational architecture		Scenario analysis, behaviour diagram	Activity Diagram
2.4	Draw a product breakdown structure tree with the system and other expected deliverables	PBS for the "extended system(end product + enabling products)"	PBS	Block Diagram
2.5	STAKEHOLDER ANALYSIS			
3	STAKEHOLDER ANALYSIS			
3.1	Define solution scope		Context analysis ???	Context Diagram
3.2	Develop 'as is' scenario from the mission statement and goals	Mission Scenario "as is"	Scenario analysis	Context Diagram
3.3	Develop 'to be' scenarios from mission statement and goals	Mission Scenario "to be"	Scenario analysis	Context Diagram
3.4	Validate scenarios with stakeholders	Validated Mission Scenarios	Stakeholders interviews	X
3.5	Identify stakeholders of the development organization (higher level systems, project management, regulatory bodies, customer, user, technology developers)	Secondary Stakeholder List ?	IDEF0	
3.6	Rank order stakeholders	Ranked Secondary Stakeholder List ?	AHP	X
3.7	Identify goals, objectives, measures of effectiveness	Secondary Stakeholder´s Goals, Objectives and	Objective tree (GQM-Goal Question Metric)	
3.8	Derive stakeholder requirements by attributing value to the measures of effectiveness	Secondary Stakeholder Requirements ?	Reqs Specification	
3.9	Derive qualification strategy for the stakeholder requirements	System Qualification Strategy	How do I assess...?	
3.10	Analyze all stakeholder requirements identifying capabilities and constraints		Requirements analysis	
3.11	Agree stakeholder requirements and qualification strategy with stakeholders	Validated Stakeholder Requirements Validated System Qualification Strategy	Stakeholder **review meetings**	
3.12	Rank order stakeholder requirements for each stakeholder	(intermediate)	AHP	X
3.13	Rank order stakeholder requirements by composing stakeholder and requirements ranking	anked Validated Secondary Stakeholder Requirements	Mathematics	X

4	ESSENTIAL FUNCTIONAL ANALYSIS			
4.1	Analyze system functional context in each of the scenarios identified	Functional Context Diagram for each scenario + DD	Context analysis, DFD	Context Diagram
4.2	Identify the states that each element in the environment of the system may assume	Environmental circumstances (in DD)	Circumstances analysis	Block Diagram
4.3	Define modes from sets of circumnstances	Modes	Circumstances analysis	Block Diagram
4.4	Perform modes analysis	STD´s for modes	**States and Modes analysis**	Block Diagram
4.5	For each mode identified, identify end-to-end timing requirements	Timing requirements, tables, graphs	**Timing analysis**	Block Diagram
4.6	For each mode identified, derive the initial essential process functions	Essential functions	Context analysis and list of events	Block Diagram
4.7	Identify essential functions from identified capabilities expressed by stakeholders	Essential functions	Requirements analysis	Block Diagram
4.8	Complete the essential functional structure	DFD´s	**Structured (Functional) Analysis**	Block Diagram
4.9	Identify system states and describe system behaviour	STD´s for states	**States and Modes analysis**	Block Diagram
4.10	Compose the essential functional architecture	Essential functional Arch. (DFDs, STDs, CFDs, DD)	Essential Functional Analysis (essential system analysis or real-time structured analysis)	Block Diagram
4.11	Associate performance requirements to each of the identified functions	Functional and Performance Requirements	Requirements analysis from stakeholder constraints	Block Diagram
5.12	Derive other non functional requirements from identified constraints	Non-Functional Requirements (all types)	Requirements analysis from stakeholder constraints	X
5.13	Write down the system requirements and corresponding qualification strategies	System Reqs Specification, verification matrix	specification techniques	X
5	HAZARD ANALYSIS			
6.1	For each mode, from the context diagram, identify hazardous events related to the circumnstances of the elements in the environment	hazardous events	Context analysis	Context Diagram
6.2	For each mode, from the context diagram, identify hazardous events related to the flows	hazardous events	Context analysis	Context Diagram
6.3	For each mode, from functional analysis, identify hazardous events related to non or malfunctioning of the identified essential functions of the system	hazardous events	Functional Hazard Analysis	X
6.4	Prioritize the identified hazards	Ranked hazards	FMECA	X
6.5	For the highest priority hazards, identify mitigation protective, preventive, corrective and detection functions, **their inputs and outputs**	hazard-related non-essential functions	FMECA	X
6.6	Include mitigation functions in the **enhanced** functional architecture derived below (7)	Part of Enhanced Functional Architecture	Functional analysis	
6	ENHANCED FUNCTIONAL ANALYSIS			
6.1	Analyzing system architecture context in each of the scenarios identified:		Architecture context diagram	Context Diagram
6.2	Identify technology non specific input processing functions	input processing functions	**DFD/CFD**	
6.3	Identify technology non specific output processing functions	output processing functions	**DFD/CFD**	
6.4	Identify technology non specific human interface functions	human interface functions	**DFD/CFD**	
7.4	Identify technology non specific enabling product functions	enabling product functions	**DFD/CFD**	
7.5	Enhance the essential functional architecture	Enhanced Functional Architecture	**DFD/CFD**	
7.6	Derive external logical interface requirements	**Reqs Specification**	Reqs specification techniques	X

7	PHYSICAL ARCHITECTURE ANALYSIS			
8.1	Compose a generic physical architecture	Physical Architecture (generic)	Architecture flow diagram, Architecture interconnect diagram	Block Diagram
8.2	Rearrange functional architecture so that functions or group of fucntions can be allocated to elements in the physical architecture		DFD, DSM	
8.3	Compose an allocatable functional architecture	Allocatable Functional Architecture	**Item 7: Extended Functional Analysis**	
8.4	Directly allocate, partition or decompose functional and non-functional requirements to each of the physical architecture elements	allocated requirements	Requirements analysis and specification, margins, budgeting	
8.5	Identify options for the implementation of elements in the generic physical architecture	implementation options for elements	Morphological chart	
10.2	Define components to be reused, to be COTS, to be developed internally, to be developed externally	Arch. Component specs	**trade-off analysis,** AID (for visual representation)	
8.6	Assess implementation alternatives for each element in the generic physical architecture	Arch. Component specs	Decision analysis, **trade-off analysis**	X
8.7	Compose a physical architecture	Physical Architecture	Architecture flow diagram, architecture interconnect diagram	Block Diagram
10.3	Define the configuration items out of the architecture block diagram	Configuration Item Identification	**Decision considering CM and project management criteria**	
8.9	Define interfaces in the physical architecture	Interface Reqs. Specification, Interface descriptions	N2 and ICDs	
8	AIT			
9.1	Define assembly, integration and verification strategy	AIV Strategy	AIT planning	X
9.2	Define assembly, integration and verification plans	AIV Plans	AIT planning	X
9.3	From requirements, define test cases	Test Cases	AIT planning	X
9.4	From test cases define verification procedures	Verification Procedures	AIT planning	X
9	COMPONENT DESCRIPTION			
10.1	Compose an architecture block diagram		ABD	Block Diagram
10.4	For COTS and reused identify further qualification required and non functional requirements	Component Requirement Specification	Requirements analysis and specification	X
10.5	For developed components, define further requirements to achieve a complete specification for development purposes.	Component Requirement Specification	Requirements analysis and specification	X

ANEXO D - TUTORIAL DA FERRAMENTA COMPUTACIONAL IBM RHAPSODY 7.6

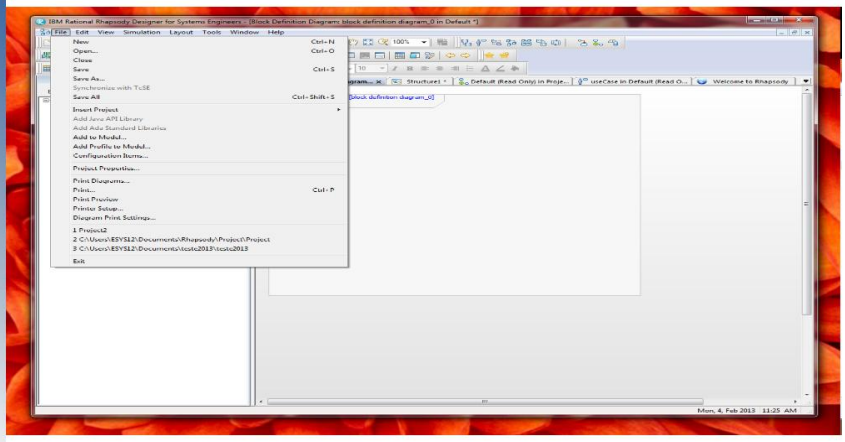
MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

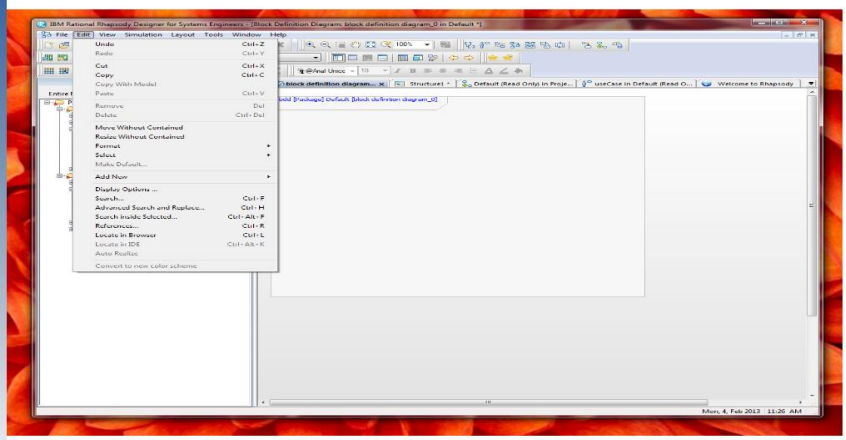
GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

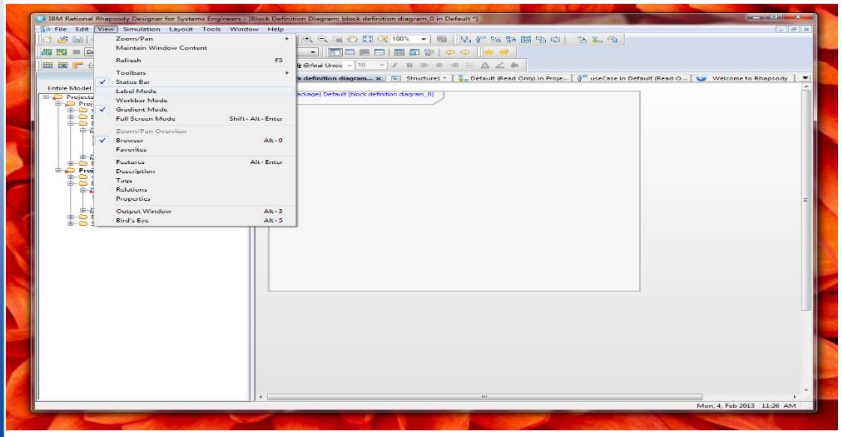
TUTORIAL DA FERRAMENTA COMPUTACIONAL IBM RHAPSODY 7.6

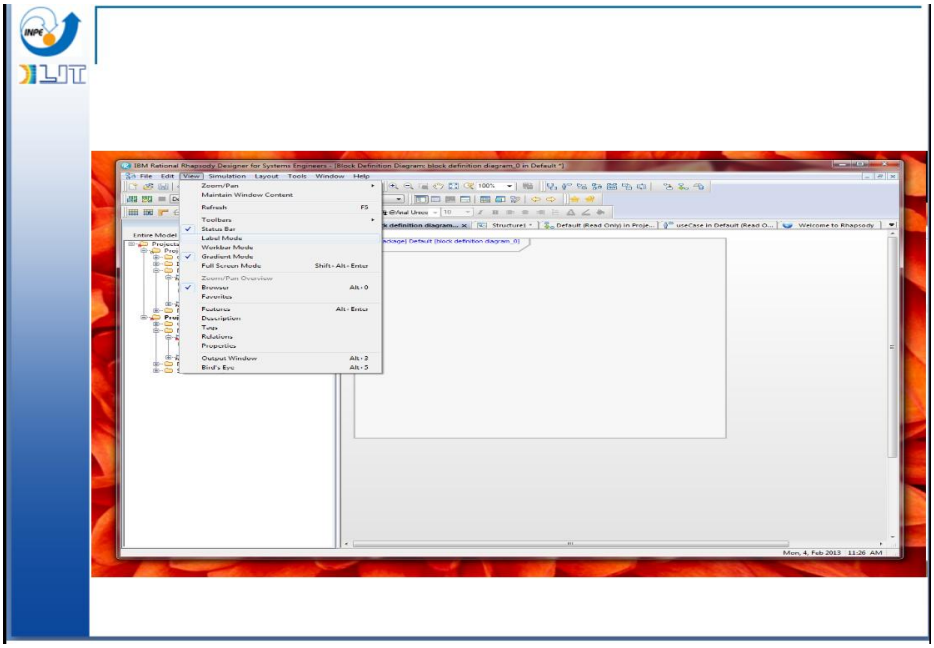
Maiara Guimarães Flausino
maiara.flausino@lit.inpe.br
Aluna de iniciação científica PIBIC do
Laboratório de Engenharia de Sistemas – LIT
Orientador: Prof. Dr. Geilson Loureiro

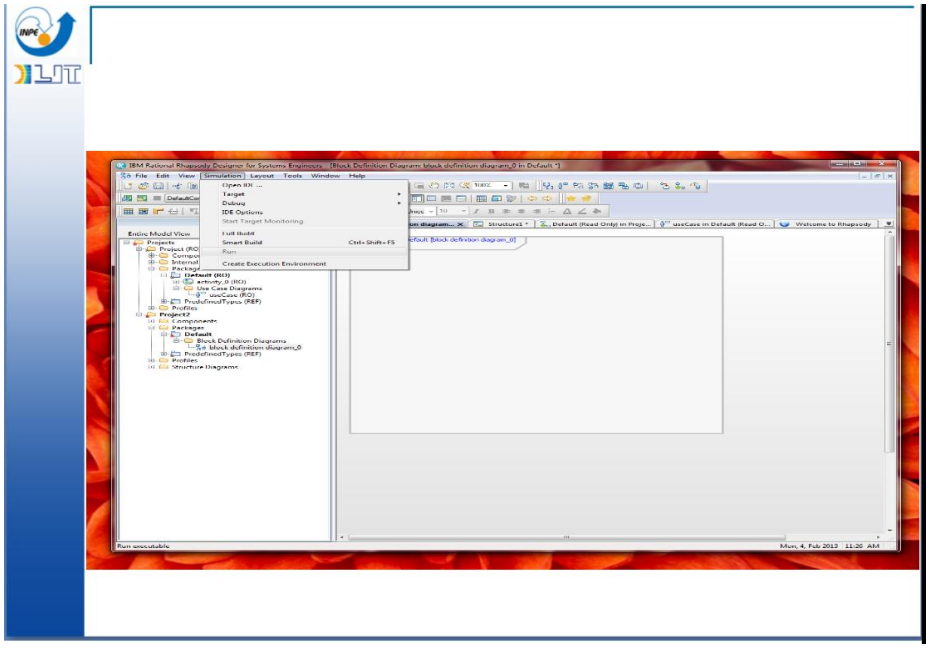
LABORATÓRIO DE INTEGRAÇÃO E TESTES

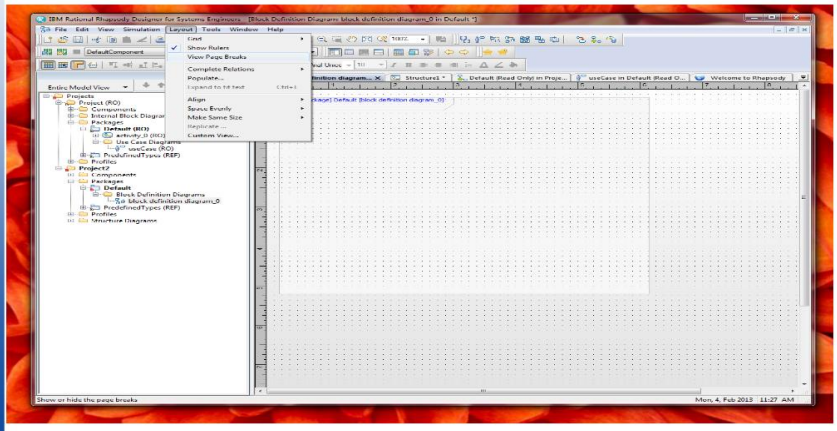


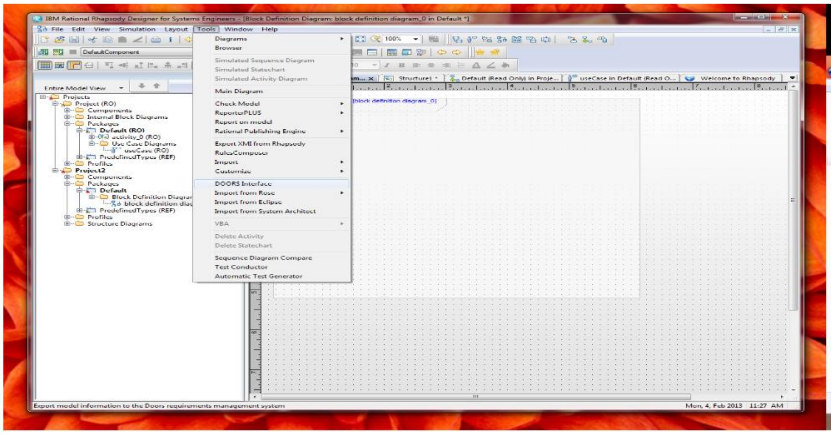


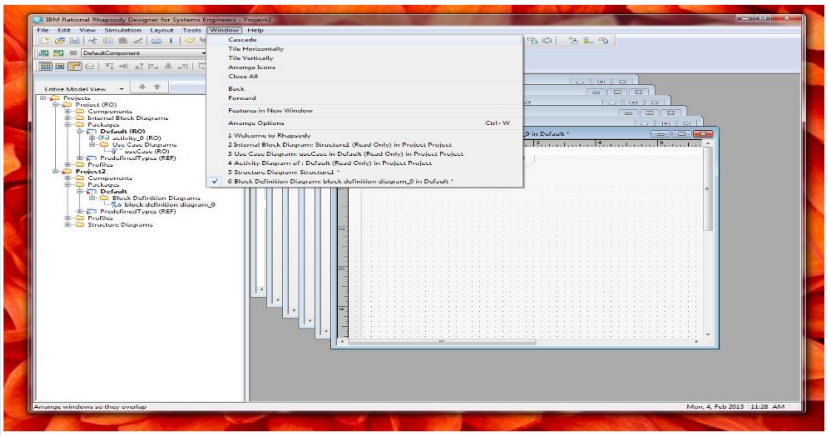


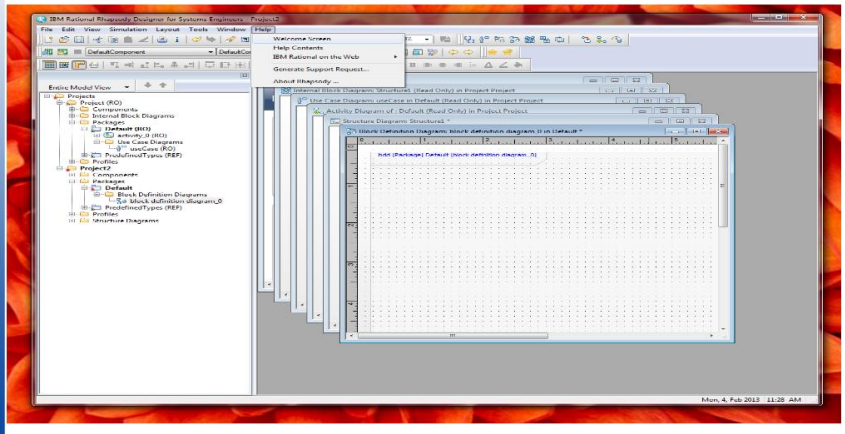


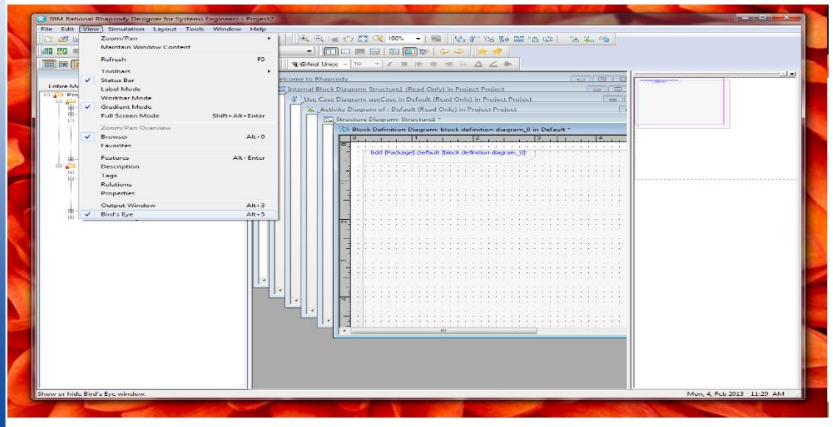


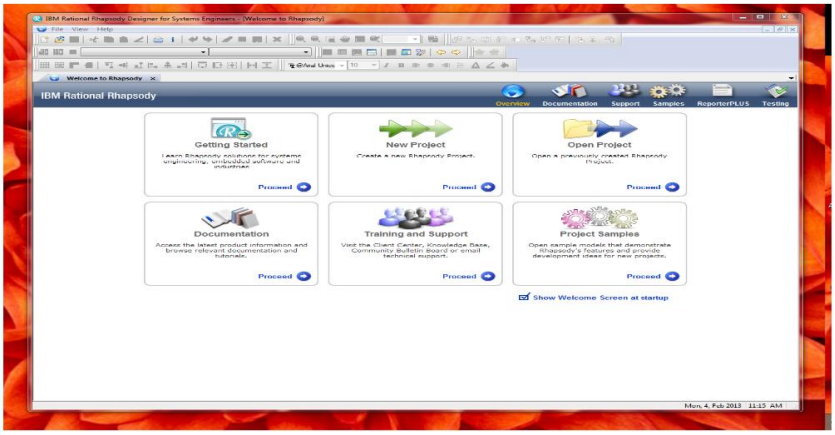


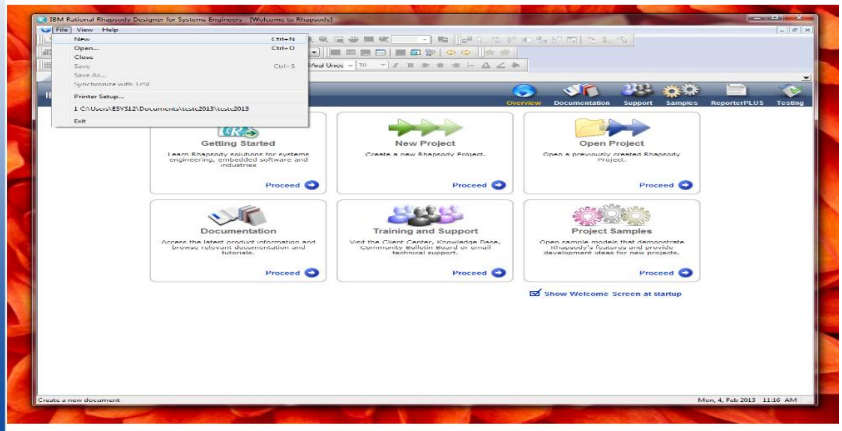


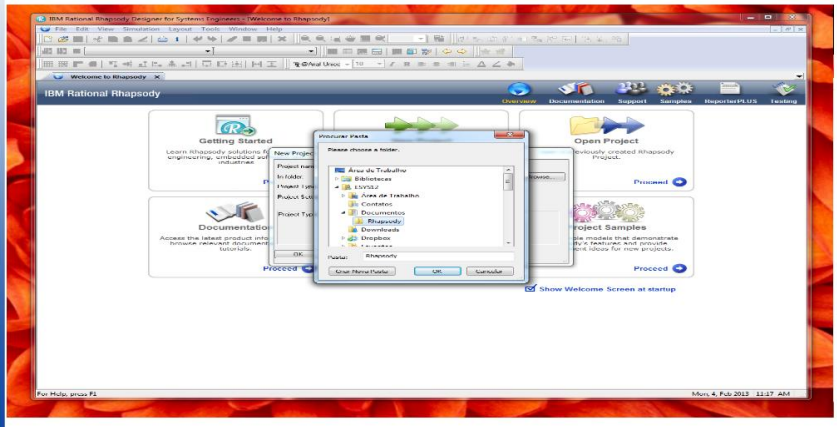


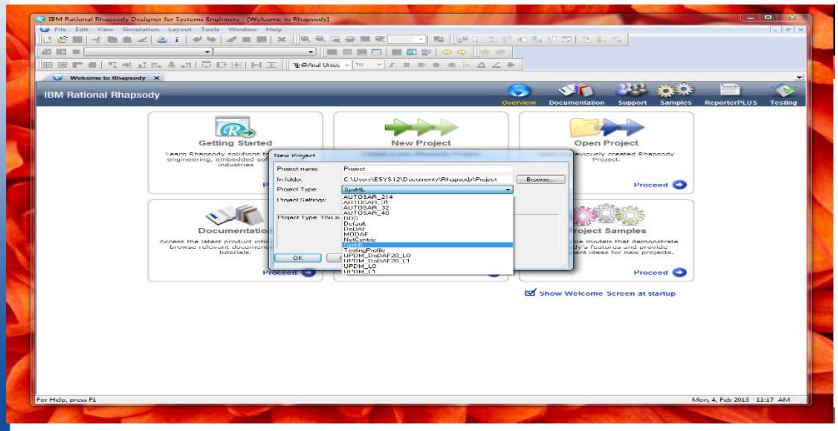


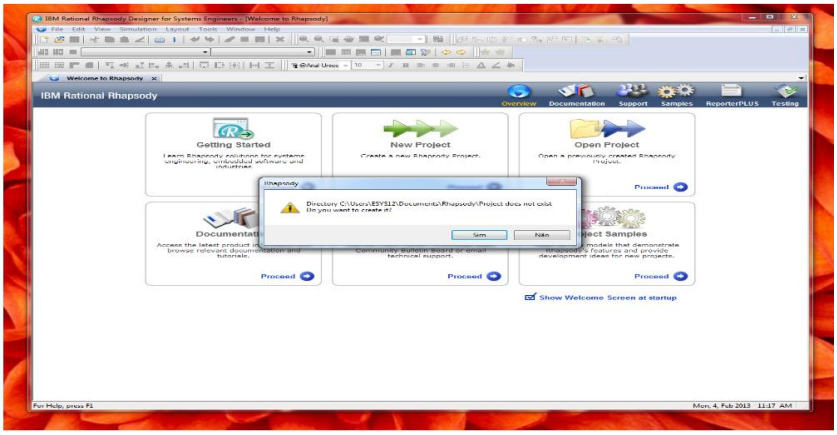


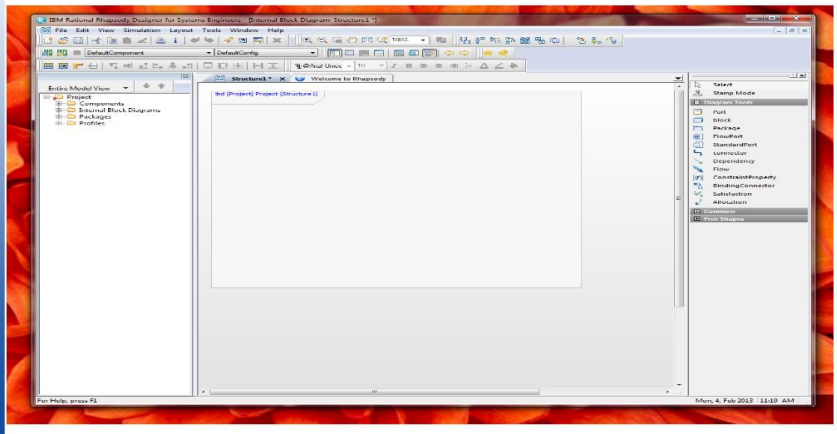

IBM Rational Rhapsody
Getting Started
New Project
Open Project
Documentation
Training and Support
Project Samples
Proceed
Show Welcome Screen at startup

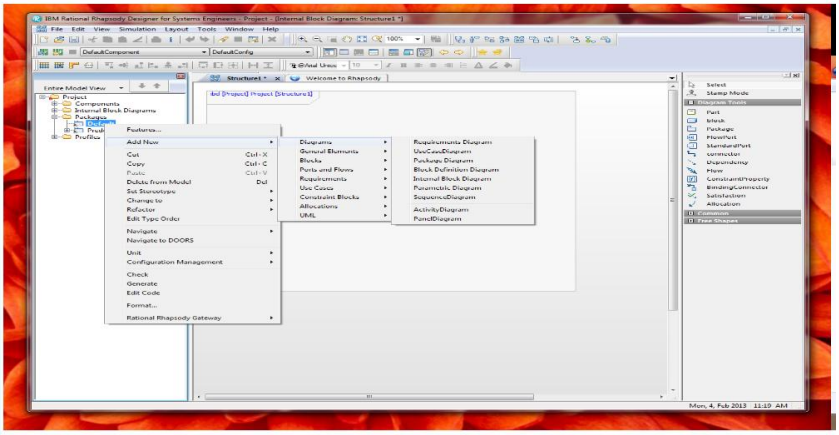

Getting Started
New Project
Open Project
Documentation
Training and Support
Project Samples
Proceed
Show Welcome Screen at startup

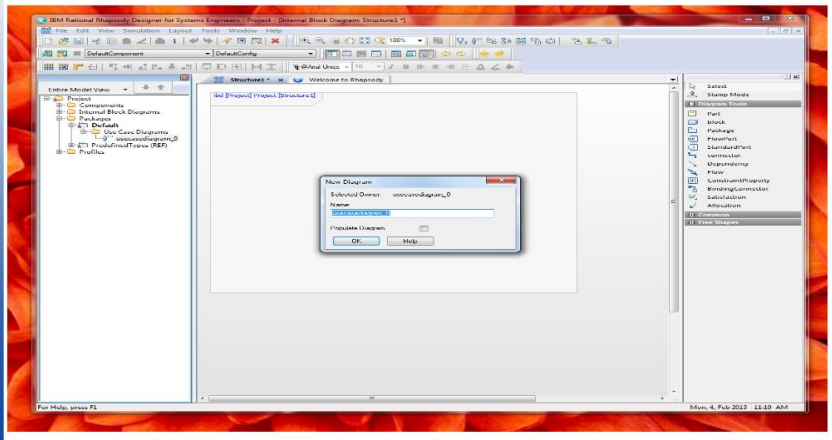

IBM Rational Rhapsody
Getting Started
Open Project
Documentation
Project Samples
Proceed
Show Welcome Screen at startup

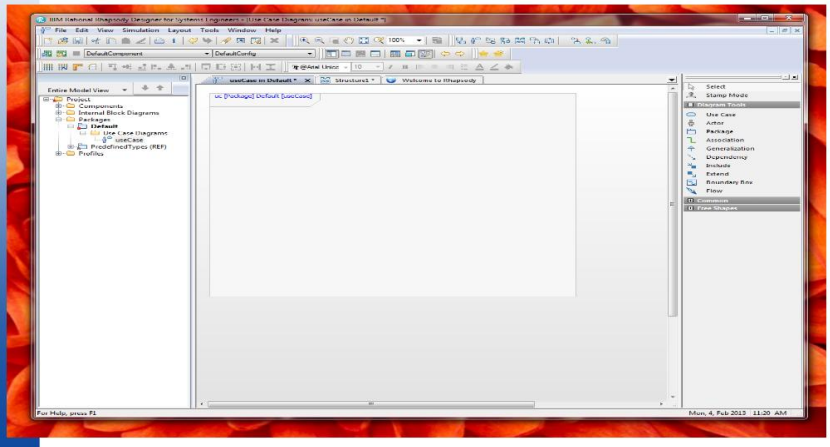

IBM Rational Rhapsody
Getting Started
New Project
Open Project
Documentation
Project Samples
Show Welcome Screen at startup

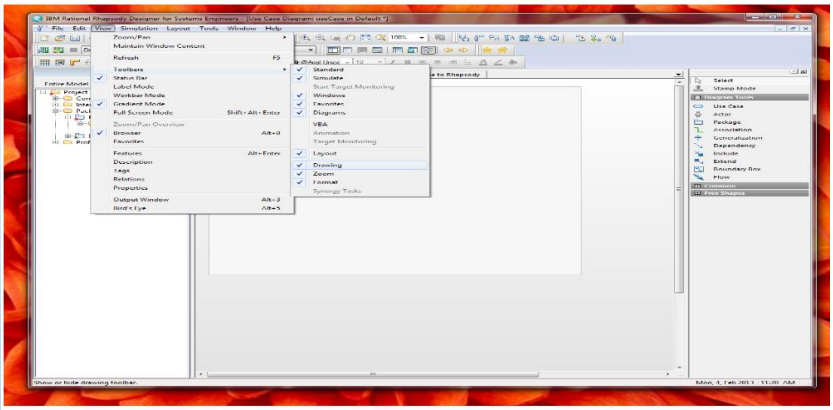

IBM Rational Rhapsody
Getting Started
New Project
Open Project
Documentation
Project Samples
Show Welcome Screen at startup

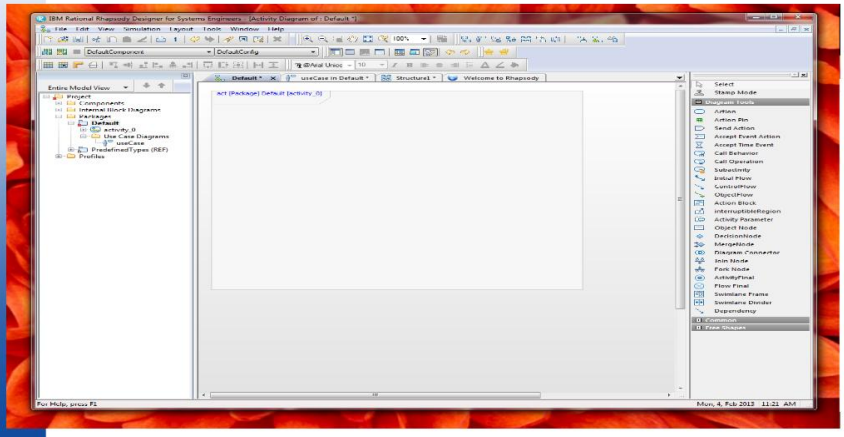


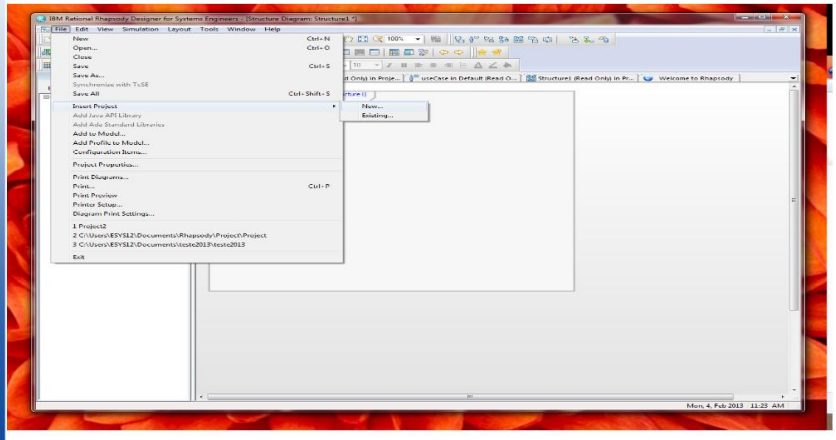


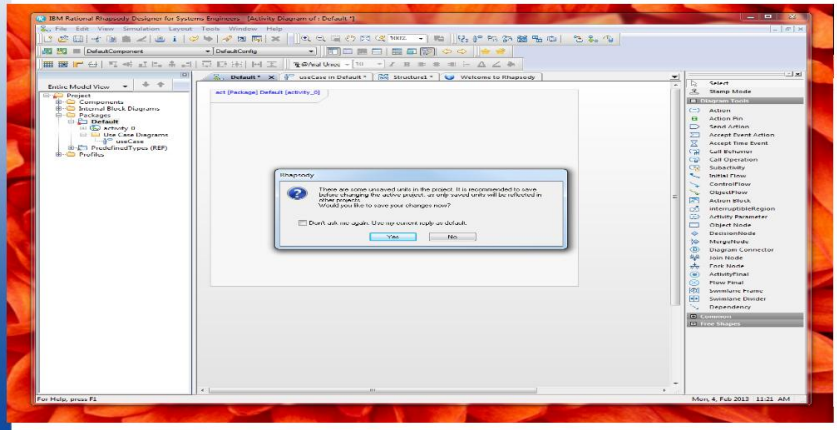


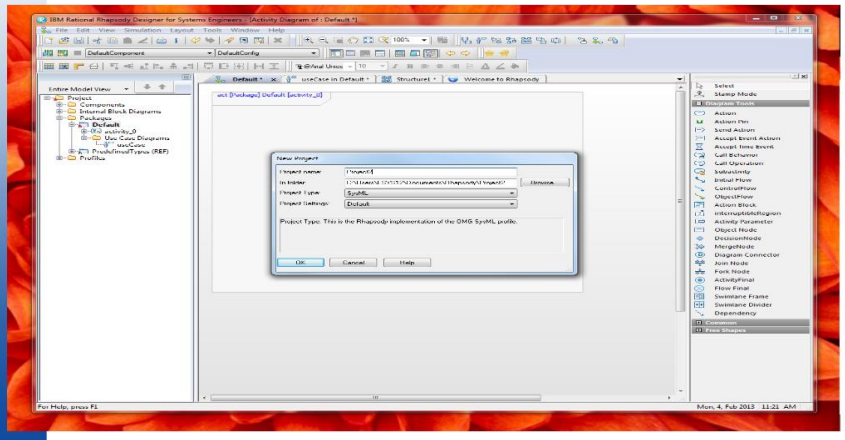


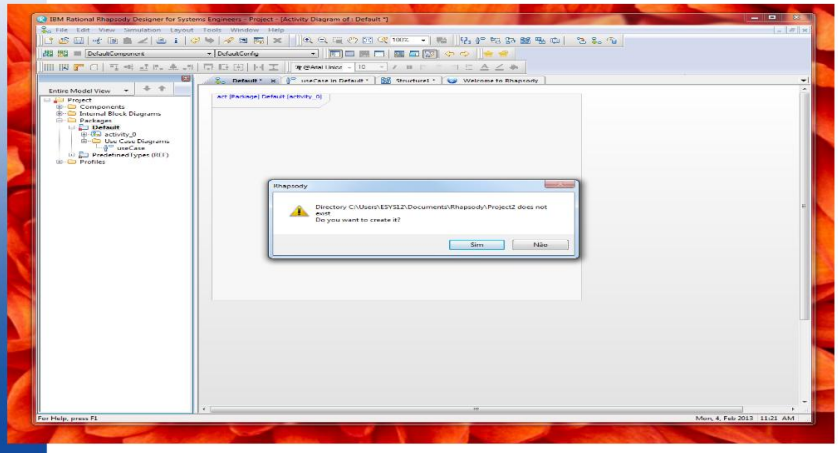


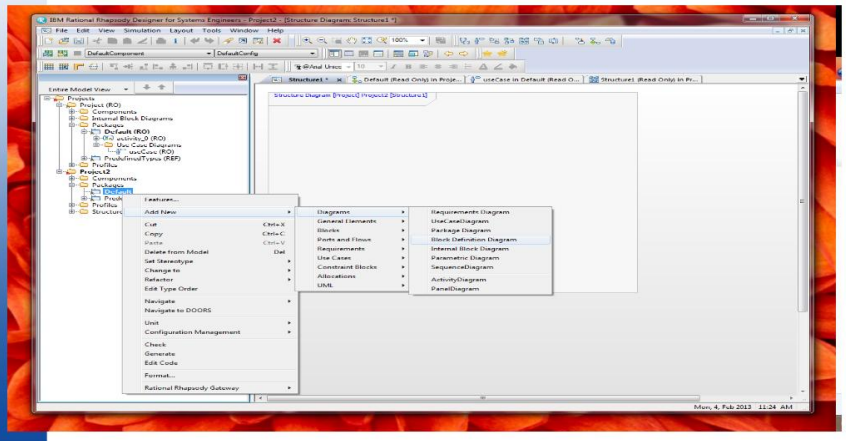


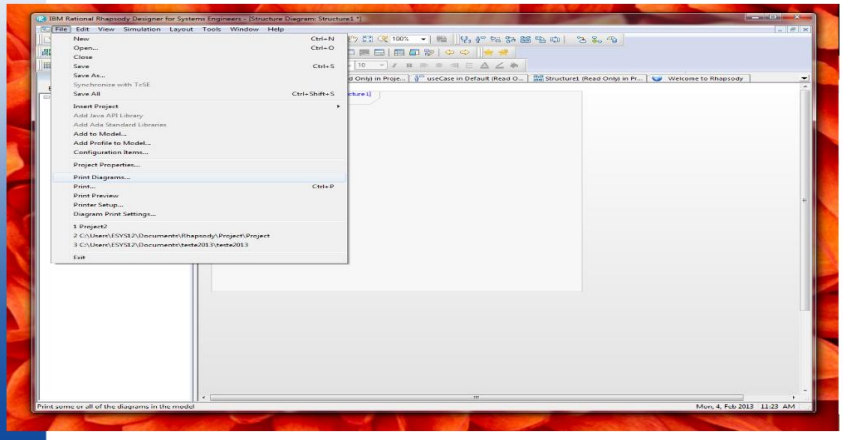


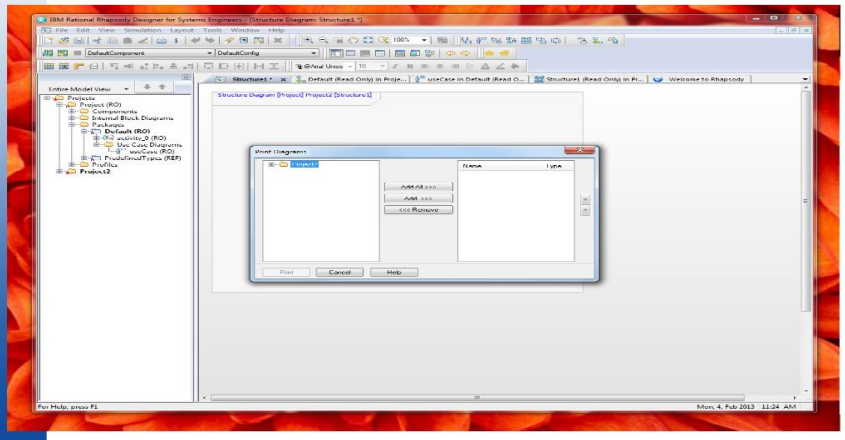


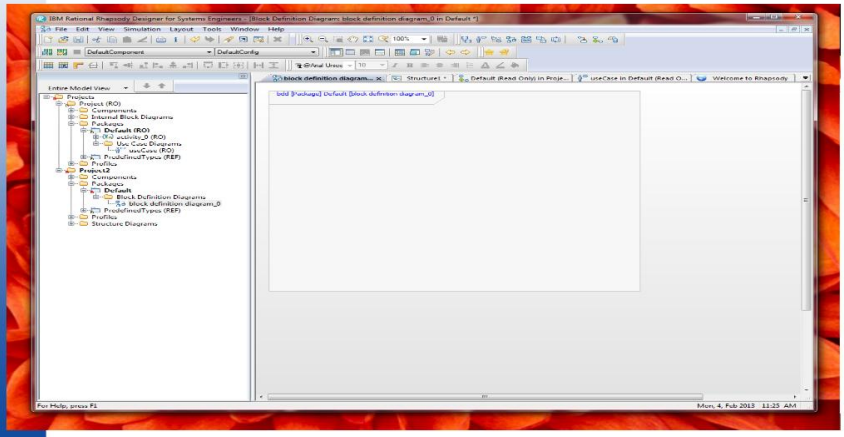


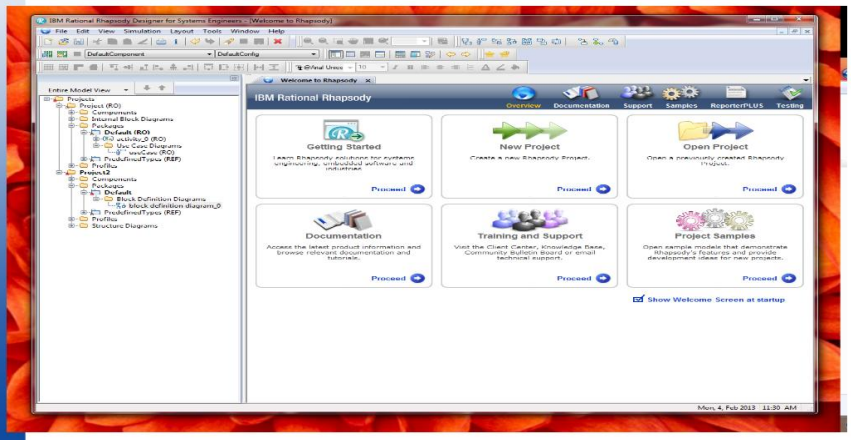


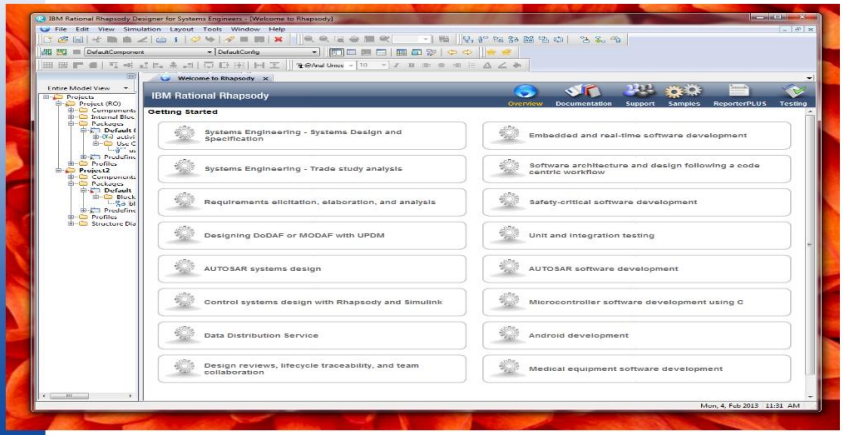


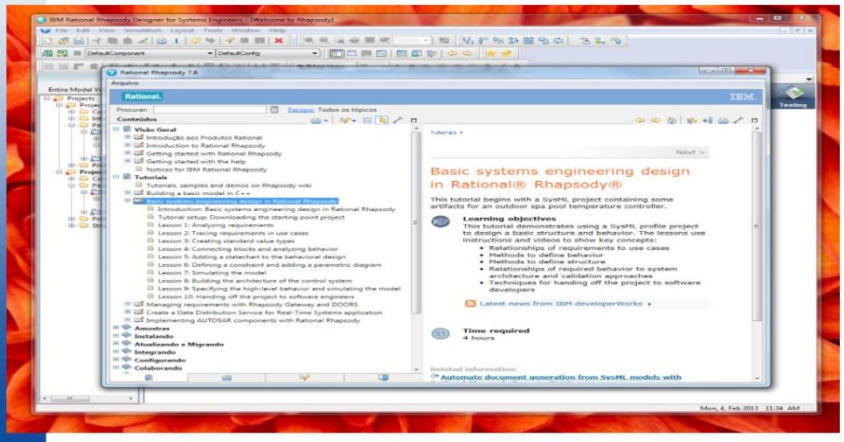


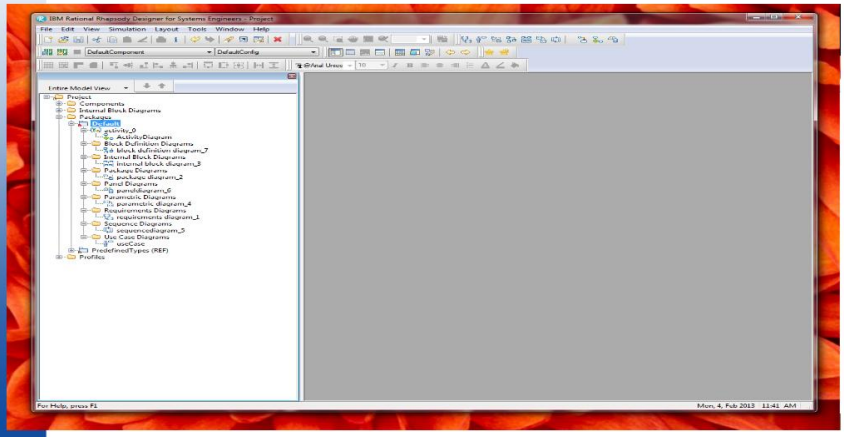


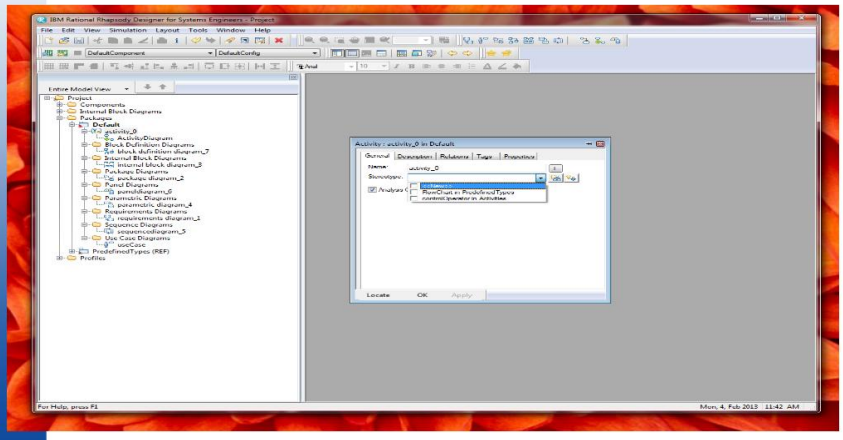


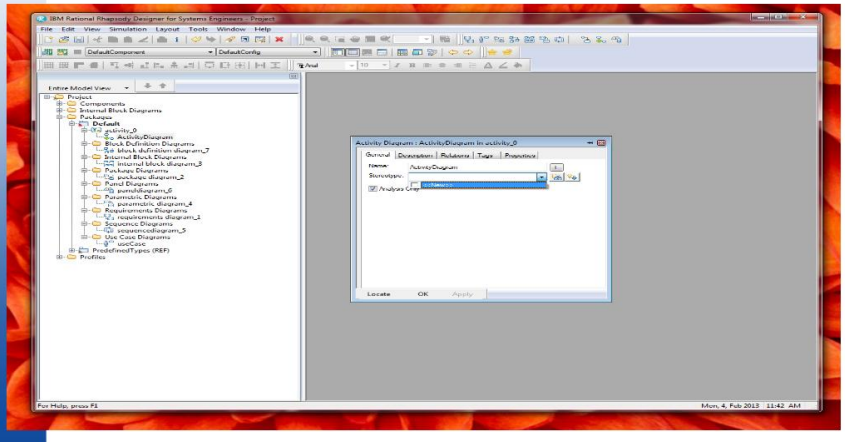

IBM Rational Rhapsody
Getting Started
New Project
Open Project
Documentation
Training and Support
Project Samples
Show Welcome Screen at startup

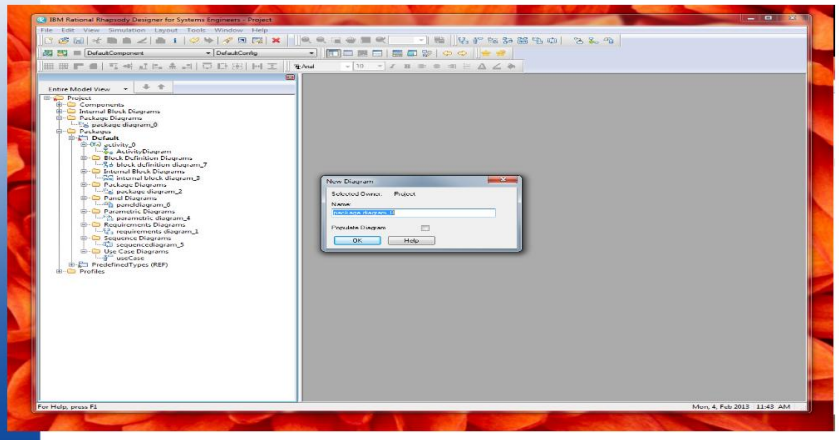

IBM Rational Rhapsody
Getting Started
Systems Engineering - Systems Design and Specification
Systems Engineering - Trade study analysis
Requirements elicitation, elaboration, and analysis
Designing DoDAF or MODAF with UPDM
AUTOSAR systems design
Control systems design with Rhapsody and Simulink
Data Distribution Service
Design reviews, lifecycle traceability, and team collaboration
Embedded and real-time software development
Software architecture and design following a code centric workflow
Safety-critical software development
Unit and integration testing
AUTOSAR software development
Microcontroller software development using C
Android development
Medical equipment software development

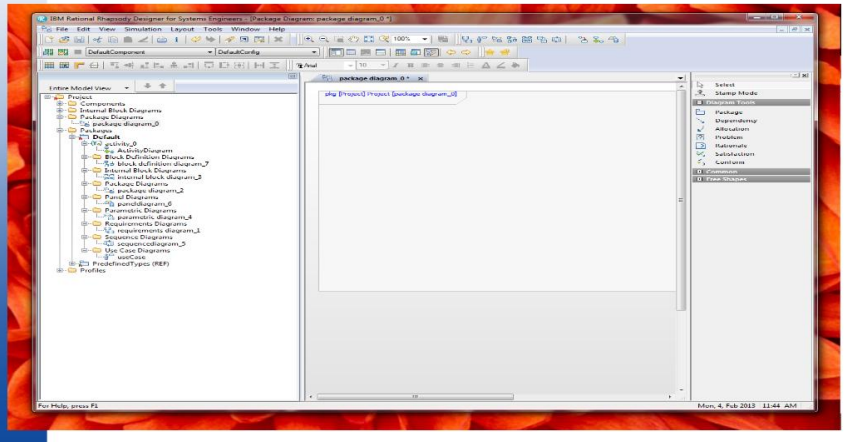

Basic systems engineering design in Rational® Rhapsody®

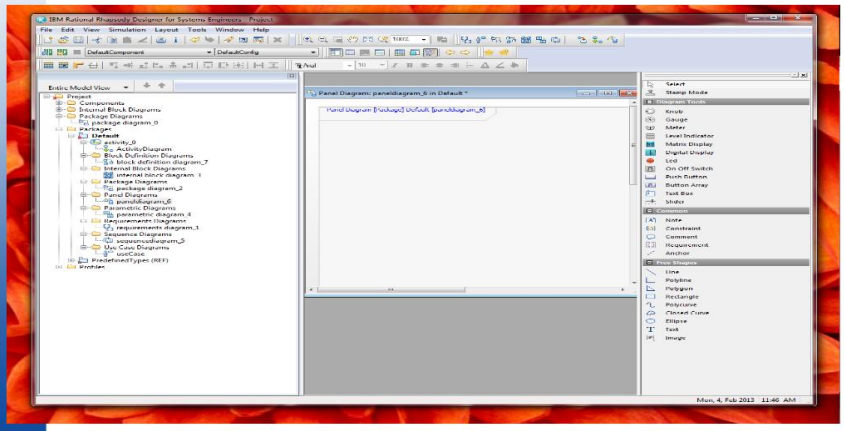


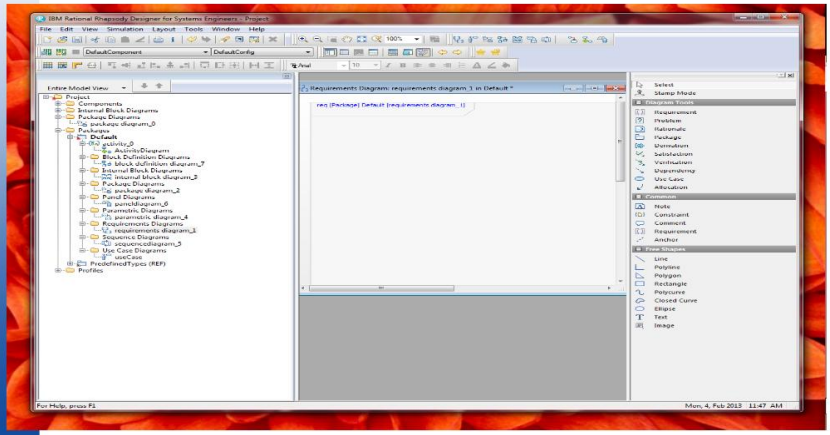


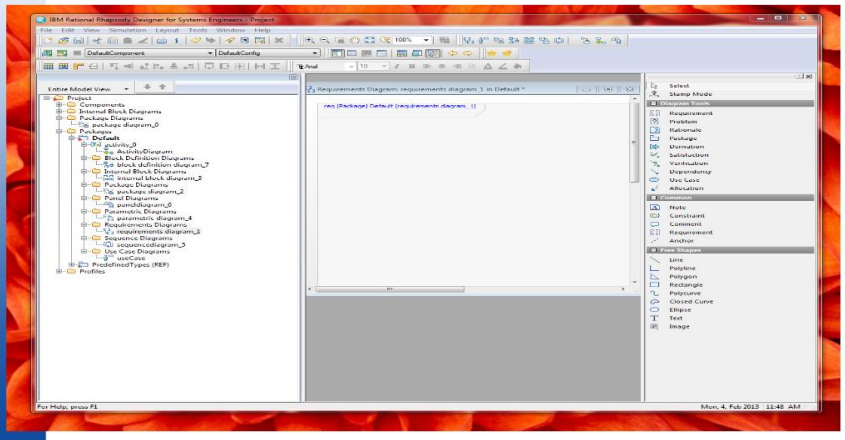


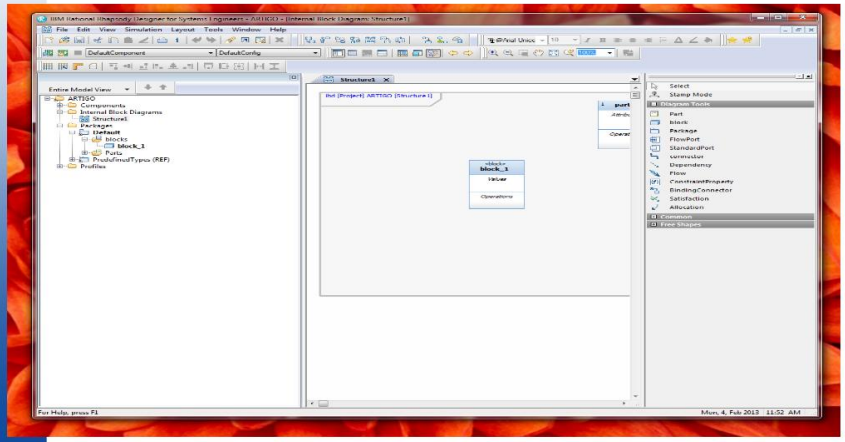


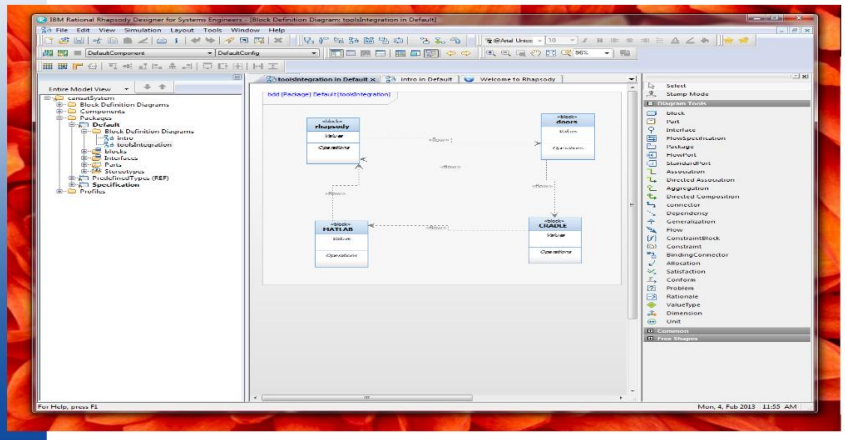


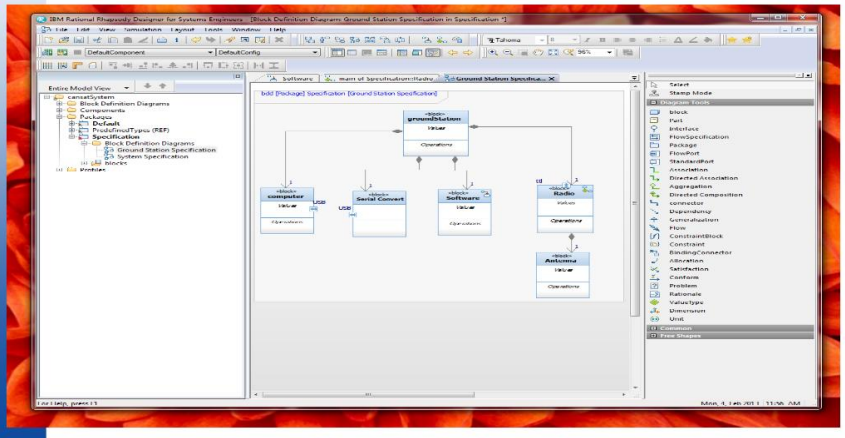


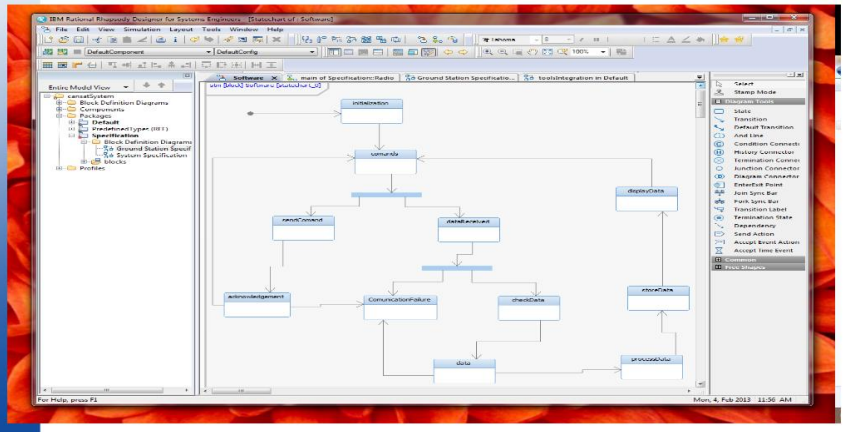


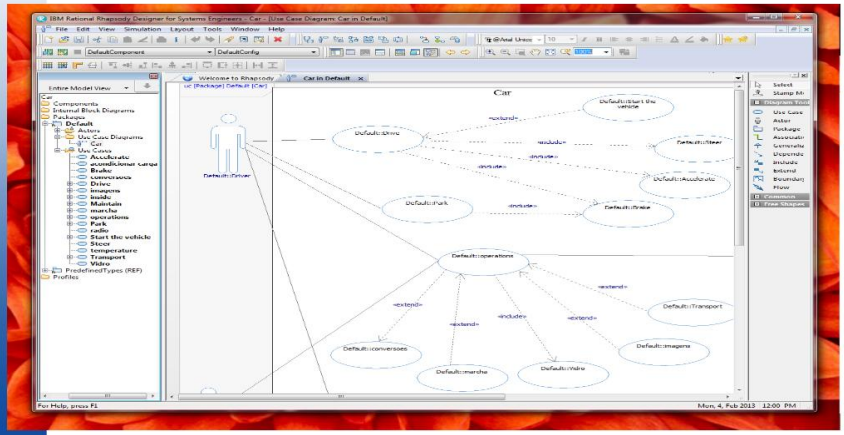


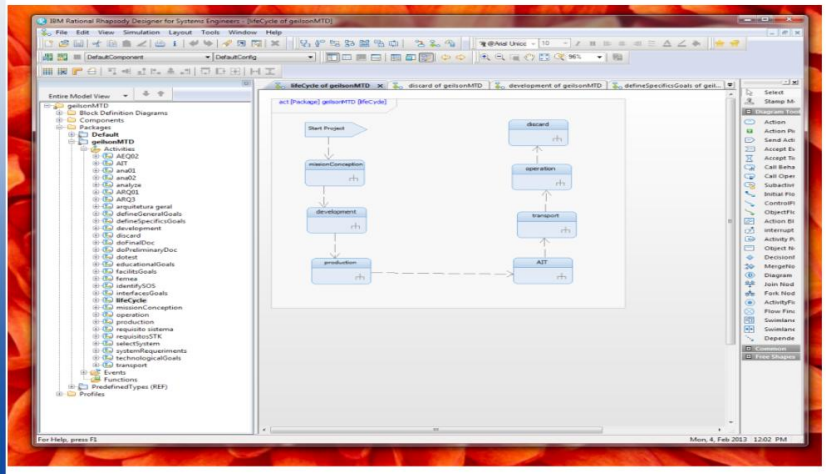


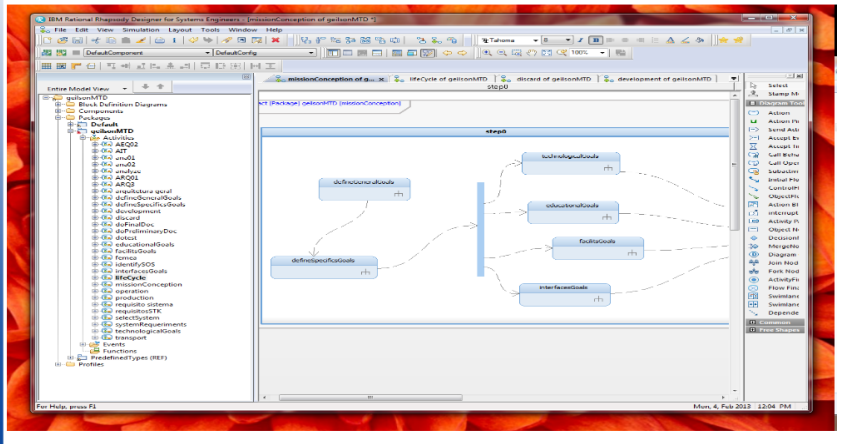












ANEXO E - Artigo científico UNIVAP 2012: SYSML PARA ENGENHARIA SIMULTÂNEA DE SISTEMAS ESPACIAIS

XVI INIC Encontro Latino Americano de Iniciação Científica
XII EPG Encontro Latino Americano de Pós Graduação
VI INIC Jr Encontro Latino Americano de Iniciação Científica Júnior

SYSML PARA ENGENHARIA SIMULTÂNEA DE SISTEMAS ESPACIAIS

Maiara Guimarães Flausino, Geilson Loureiro

Instituto Nacional de Pesquisas Espaciais/Laboratório de Integração e Testes, Avenida dos Astronautas, 1758, Jardim da Granja, CEP 12.227-010 - São José dos Campos – SP
maiara.flausino@lit.inpe.br, geilson@lit.inpe.br

Resumo- Este trabalho tem por objetivo analisar, compreender e aplicar o método de engenharia simultânea de sistemas proposto por Loureiro (1999) em sua Tese de Doutorado. Para isso, foi utilizada a *Systems Modeling Language (SysML)* para modelar um sistema espacial. O produto espacial escolhido para ser modelado foi o Sistema CANSAT, composto por um Foguete, uma Estação Terrena e um Picossatélite, o qual foi utilizado como exemplo em situação de sala de aula. Este método consiste no desenvolvimento simultâneo do produto e das organizações que implementam os processos do ciclo de vida de um sistema ao longo de suas etapas, quais sejam, análise de *stakeholders*, análise de requisitos, análise funcional e arquitetura de sistemas.

Palavras-chave:*SysML*, Engenharia Simultânea de Sistemas, Modelagem Gráfica, Sistemas Complexos
Área do Conhecimento: ENGENHARIAS

Introdução

A engenharia simultânea de sistemas requer uma visão global de todas as fases do ciclo de vida de um produto. Evidência disto é a evolução dos conceitos desta abordagem adotados pelas indústrias automobilísticas e aeroespaciais no sentido de obter uma solução balanceada que considere as variáveis, como o tempo de desenvolvimento, o custo dos processos do ciclo de vida, o gerenciamento de risco, o desempenho de um produto e, ainda, atenda aos seus requisitos (LOUREIRO, 1999).

A evolução citada anteriormente consiste em analisar os *stakeholders*, os requisitos, o conceito funcional e a arquitetura de implementação, simultaneamente, para o produto, seus processos de ciclo de vida e suas organizações, atuando em todas as camadas da estrutura de desenvolvimento de um sistema complexo (LOUREIRO, 2011).

A técnica que facilita o entendimento da complexidade de um sistema é a modelagem, sendo ela um dos pilares da engenharia simultânea de sistemas, especialmente a modelagem gráfica, a qual permite ao profissional ter uma visão do todo, bem como identificar cada relacionamento existente num dado sistema.

A modelagem de um sistema possibilita realizar o rastreio dos requisitos, os quais futuramente serão validados com os *stakeholders*, contribuindo de forma positiva ao desenvolvimento do projeto.

Para acompanhar a evolução técnico-científica, encontra-se em desenvolvimento uma nova linguagem de modelagem gráfica, a *Systems Modeling Language (SysML)*, a qual tem como objetivo principal padronizar os modelos de sistemas complexos.

Diante deste contexto, este trabalho tem por objetivo analisar, compreender e aplicar o método de engenharia simultânea de sistemas proposto por Loureiro (1999) em sua Tese de Doutorado.

Este trabalho está organizado da seguinte forma: seção dois apresenta a metodologia, em seguida é apresentado os conceitos teóricos que deram base para o desenvolvimento deste trabalho, na terceira e quarta seção serão expostos os resultados e as discuções e por último será apresentada a conclução desta pesquisa.

Metodologia

A metodologia utilizada neste estudo envolveu diferentes momentos. No início, com base na análise crítica da literatura científica pesquisada, foi feita a extração de conceitos do referencial teórico sobre a *Systems Modeling Language (SysML)* e a engenharia simultânea de sistemas. Estes ofereceram os fundamentos a partir dos quais o Sistema CANSAT foi modelado.

A realização deste trabalho ocorreu entre agosto de 2011 e julho de 2012. Durante esse período de pesquisas e estudo continuo, através da revisão bibliográfica, leitura de artigos e revistas científicas da área, sobre a linguagem *SysML*, foram investigadas suas características, seus diagramas, suas utilidades e suas perspectivas de uso.

Durante o processo de aprendizagem, foi realizada uma vasta pesquisa sobre as ferramentas, que servem para modelar na linguagem *SysML*, sendo que o uso da cada *software* deu-se com a licença gratuita para testes.

Depois da fase de pesquisa e consolidação dos conhecimentos adquiridos, foi dado início ao desenvolvimento do *template* que tem por objetivo contribuir ao entendimento da aplicação do método de engenharia simultânea de sistemas proposto por Loureiro (1999).

Desta forma, para colocar em prática o aprendizado da linguagem *SysML* e realizar o *template*, foi necessário o treinamento da ferramenta *IBM Rational Rhapsody*, o qual se deu através de manuais de usuários, tutoriais virtuais e também da participação em foruns *online*.

E como resultado apresentado neste trabalho, o *software* utilizado foi o *IBM Rational Rhapsody*, versão completa, disponível no Laboratório de Engenharia de Sistemas (LSIS) do Laboratório de Integração e Testes (LIT).

Systems Modeling Language (SysML)

Devido a evolução das tecnologias dos diversos setores, houve um grande aumento na complexidade de seus produtos. Diante desta situação, foi proposta uma nova linguagem de modelagem gráfica a *Systems Modeling Language (SysML)*, a qual está sendo desenvolvida pelo *OMG (Object Management Group)* e pelo *INCOSE (International Council on Systems Engineering)*.

A linguagem *SysML* tem como propósito geral analisar, especificar, projetar, verificar e validar sistemas complexos, sendo que nesses sistemas podem estar incluídos *hardwares*, *softwares*, pessoas, facilidades e outros elementos (FRIEDENTHAL, 2009).

A linguagem *SysML* derivou-se da *Unified Modeling Language (UML)* usada exclusivamente para modelar *softwares* de alto desempenho, sendo que quatros diagramas da linguagem *UML* permaneceram com as mesmas características e os outros cinco foram criados e adaptados para serem capazes de modelar os sistemas complexos na linguagem *SysML*, como mostra a Figura 1, formando os nove diagramas da desta linguagem. Ela já permite a descrição consistente de sistemas entre os diferentes tipos de profissionais do mesmo projeto, como, engenheiros de software, mecânicos e eletricistas.

Figura 1 - Diagramas da linguagem *SysML*.
Fonte: Elaborado pelos autores.

A partir de 2007, ferramentas computacionais que implementam esta linguagem passaram a estar disponíveis no mercado mundial.

A linguagem *SysML* considera as características no domínio de um sistema em diferentes níveis de abstração, possibilitando a rastreabilidade de requisitos, a modelagem de sua estrutura e de seu comportamento, bem como o formalismo paramétrico para especificar suas equações.

Desta forma, permite modelar o ciclo de vida completo de um sistema de alta tecnologia em projetos relacionados à engenharia simultânea de sistemas desde a sua fase de projeto até a sua análise e melhoria continua, sendo assim, com a linguagem *SysML* é possível modelar e documentar todos processos propostos por Loureiro (1999), garantindo a visão sistêmica do seu método.

Engenharia Simultânea de Sistemas

A engenharia simultânea de sistemas, abordagem usada para modelar o *template* deste trabalho, antecipa os requisitos dos processos do ciclo de vida de um sistema (Figura 2) para as etapas iniciais do seu desenvolvimento. Essa antecipação facilita a implementação de um produto diminuindo os riscos e os custos envolvidos e aumentando a chance de se obter sucesso no final do projeto, já que setenta por cento do custo do ciclo de vida de um produto é empregado no início do seu desenvolvimento (LOUREIRO, 2011).

Figura 2 - Ciclo de vida do produto e seus cenários.
Fonte: Loureiro (Notas de aula, 2011).

Resultados

Este trabalho aplicou a abordagem de engenharia simultânea de sistemas, usando a linguagem *SysML*, na construção de um *template*. Esta abordagem proposta por Loureiro (1999) deve ser aplicada em todos os níveis de abstração de um produto complexo.

Para realizar a análise do Sistema, garantindo a compreensão de todos os seus subsistemas e elementos foi necessário considerar dois tipos de pensamentos: a) Hierárquico, o qual organiza os elementos em cada nível de um sistema. b) Sistêmico, o qual ajuda na análise funcional, uma vez que um sistema depende das relações entre seus componentes.

Estes dois tipos de pensamento devem ser o mais abstrato possível, isto é, deve-se pensar de maneira funcional, por exemplo, muitas vezes o *stakeholder* pede um manômetro, mas o que ele precisa realmente é de um simples objeto que meça a pressão de um fluido.

A linguagem *SysML* segue o paradigma de orientação a objeto, permitindo criar, segundo a norma da própria linguagem, novos estereótipos de elementos, de relações e de diagramas. Essa característica faz com que ela não seja prescritiva.

Usando esta propriedade da linguagem, neste trabalho foi criado o Diagrama de Contexto, a partir do Diagrama de Definição de Blocos, já existente na linguagem *SysML*, para identificar os *stakeholders* do Sistema CANSAT.

O Diagrama de Contexto pode ser usado também para representar tanto o contexto funcional como o contexto físico de um produto. Foi criado este diagrama dada a importância dos *stakeholders* no processo de engenharia simultânea de sistemas.

Por isso, é importante valorizar os dados que o *stakeholder* oferece em suas respostas. E para extrair o máximo de dados é importante perguntar várias vezes sobre o mesmo assunto, de maneira sistemática, pois é a partir desse processo que a necessidade do *stakeholder* será capturada.

Apresenta-se a seguir as principais características deste novo estereótipo de diagrama, o Diagrama de Contexto: a) Apresenta um conjunto de elementos relacionados para um determinado propósito, como especificar um *stakeholder*. b) Define o ambiente em que o sistema vai pertencer, demonstrando as suas características. c) É composto por fluxos de material, energia e informação, que mostram as interfaces entre o sistema e as entidades externas. d) Permite identificar os limites dos processos, as áreas envolvidas com o processo e os relacionamentos com outros processos e elementos externos à empresa (ex.: *stakeholders*).

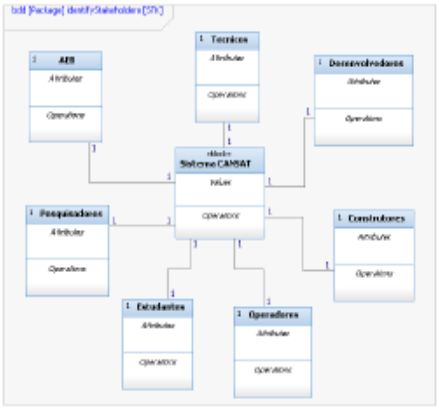


Figura 3 - Stakeholders do Sistema CANSAT.
Fonte: Elaborado pelos autores.

Discussão

Com a duração de um ano de atividades, esta pesquisa teve como resultado a primeira versão do *template* da modelagem do Sistema CANSAT. Muitas adaptações foram e serão ainda necessárias para a completa realização deste *template*, pois o método utilizado foi inicialmente modelado com base na Análise Estruturada, ou seja, os modelos originais refletem diretamente a linguagem estruturada, separando os registros (dados) das funcionalidades (procedimentos e funções).

O *template* desenvolvido foi realizado no *software IBM Rational Rhapsody*. Foi necessário usar uma ferramenta computacional devido à organização de sua hierarquia, pois o nível de detalhamento do produto pode ser elevado, ao apresentado neste trabalho, chegando a especificar os seus componentes, isto é, conhecer todo o conjunto de interfaces de um sistema.

Os diagramas modelados através de uma ferramenta computacional contêm vantagens em relação a uma simples figura, como, a geração de código automaticamente e a substituição de documentos (páginas descritivas) da área técnica da engenharia simultânea de sistemas em modelos gráficos, o qual servirá como base para realizar simulações dos modelos.

Com a modelagem apresentada, é possível visualizar o todo de maneira organizada, facilitando a análise do sistema, tanto da parte comportamental quanto da estrutural, além dos requisitos e parâmetros do produto espacial.

É importante ressaltar que o resultado apresentado neste trabalho abrangeu a modelagem, usando a linguagem *SysML*, do segmento da Estação Terrena do Sistema CANSAT.

Sistema CANSAT

O modelo apresentado na Figura 4 ilustra o Sistema CANSAT e seus subsistemas (Foguete, Estação Terrena e Picossatélite). A Figura 4 foi modelada, a partir do Diagrama de Definição de Blocos, já que visa representar a parte estrutural do Sistema, o qual representa seus componentes e seus relacionamentos.

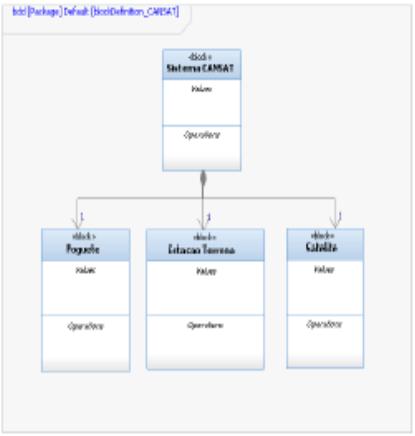


Figura 4 - Sistema CANSAT.
Fonte: Elaborado pelos autores.

Processos da Engenharia Simultânea de Sistemas da Estação Terrena

A Figura 5 mostra a organização das etapas de engenharia simultânea de sistemas por pacotes. Pois, na linguagem *SysML*, o Diagrama de Pacotes deve ser usado para organizar a estrutura de um sistema ou organização, antes da criação de seus diagramas, para assegurar que a modelagem seja fiel à realidade observada.

Sintetizar é um processo lento e complexo, sendo que construir modelos é fazer sínteses da realidade. Em razão disso, o problema deve ser estudado e estruturado logo no início do projeto.

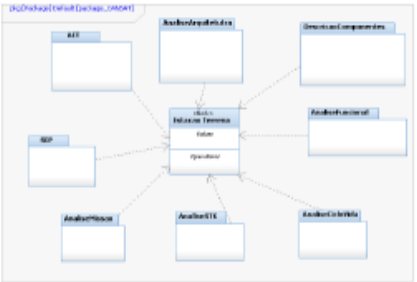


Figura 5 – Processos da Engenharia Simultânea de Sistemas da Estação Terrena.
Fonte: Elaborado pelos autores.

Estação Terrena e seus Subsistemas

No diagrama da Figura 6 foi dado um *zoom* no bloco da Estação Terrena da Figura 4, mostrando os seus subsistemas. Neste caso, foi usado um Diagrama de Bloco Interno, o qual permite um maior detalhamento em relação ao bloco apresentado no Diagrama de Definição de Blocos, expressando suas partes e seus relacionamentos, isto é, demonstra a hierarquia entre o Sistema e seus subsistemas.

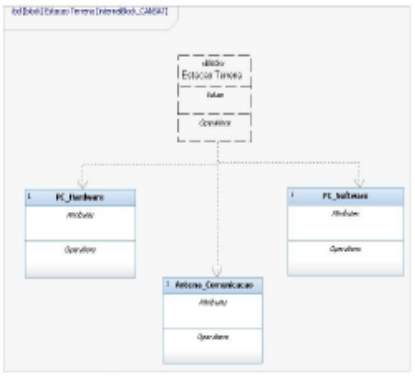


Figura 6 - Estação Terrena e seus Subsistemas.
Fonte: Elaborado pelos autores.

Casos de Uso da Estação Terrena

A fim de identificar os requisitos e distinguir as funções básicas de um sistema, o Diagrama do Caso de Uso (Figura 6) é o mais indicado, pois possui uma sintaxe simples. Geralmente este tipo de diagrama é usado na fase de validação de um sistema com os *stakeholderes*.

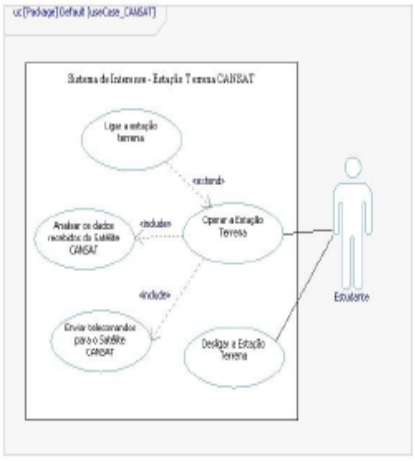


Figura 7 - Casos de Uso da Estação Terrena.
Fonte: Elaborado pelos autores.

Ciclo de Vida da Estação Terrena

A seguir será apresentado o ciclo de vida da Estação Terrena do Sistema CANSAT (Figura 8). A modelagem seguiu o pensamento sistêmico e hierárquico. Com isso, os cenários foram modelados em um Diagrama de Atividades, pois expressa o comportamento do produto espacial, uma vez que o comportamento denota dinâmica de um elemento, por exemplo, a operação de um sistema. A modelagem dos cenários é importante, porque é através dela que se pode derivar mais requisitos de sistema.

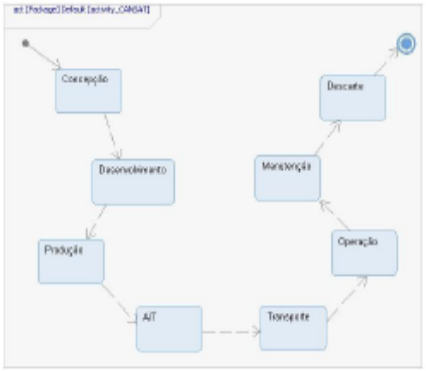


Figura 8- Ciclo de Vida da Estação Terrena.
Fonte: Elaborado pelos autores.

Estados da Estação Terrena

O Diagrama de Máquina de Estados descreve o comportamento de um sistema (Figura 9). O estado de um sistema é um conjunto de circunstâncias ou atributos que o caracterizam em certo período, sendo que a transição de estados é a mudança de estado para outro.

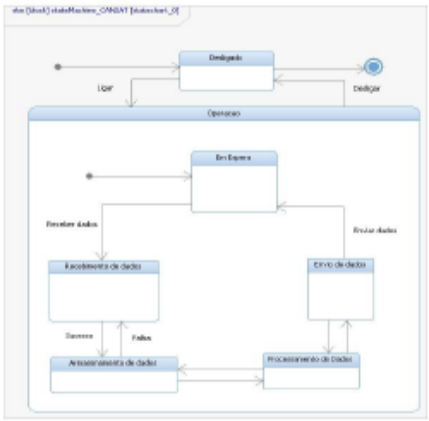


Figura 9 - Estados da Estação Terrena.
Fonte: Elaborado pelos autores.

Operação da Estação Terrena

O Diagrama de Sequência utilizado para modelar o cenário de operação do ciclo de vida da Estação Terrena (Figura 10) descreve o fluxo de controle entre o operador, neste caso o estudante, e o Picossatélite. Este diagrama evidencia quais são os eventos que ocorrem em um determinado processo, mostrando os elementos envolvidos nos disparos dos eventos e a ordem em que são realizados.

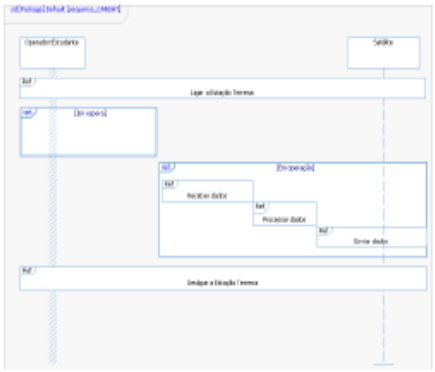


Figura 10 - Operação da Estação Terrena.
Fonte: Elaborado pelos autores.

Parâmetros da Estação Terrena

A Figura 11 ilustra a equação de fluxo de dados trocados entre o Picossatélite e a Estação Terrena do Sistema CANSAT, através do Diagrama Paramétrico, o qual mostra os atributos sobre as propriedades do Sistema, auxiliando em sua análise.

Os modelos paramétricos capturam as propriedades restritivas do sistema, para serem analisadas por ferramentas de análise apropriadas. As restrições são apresentadas por equações, cujos parâmetros estão vinculados às propriedades do sistema (FRIEDENTHAL, 2009).

Figura 11 - Parâmetro do fluxo de dado da Estação Terrena.
Fonte: Elaborado pelos autores

Requisitos da Estação Terrena

O processo de elicitação e análise de requisitos recebe na entrada os dados fornecidos pelo *stakeholder* e fornece como saída um conjunto de requisitos documentados e priorizados de um sistema. A modelagem de requisitos pode ser feito na linguagem *SysML*, pois esta dá suporte a este processo.

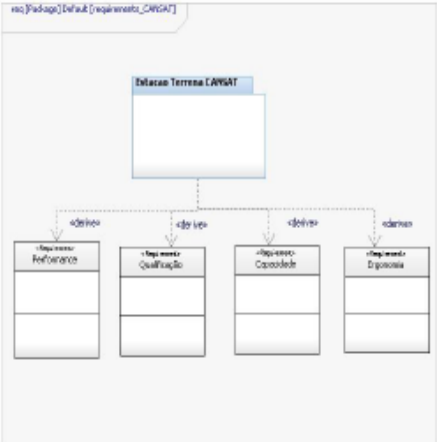


Figura 12 - Requisitos da Estação Terrena.
Fonte: Elaborado pelos autores.

Conclusão

Este trabalho realizou a modelagem da primeira versão do *template*, utilizando a linguagem *SysML*, de um produto espacial seguindo o método de engenharia simultânea de sistemas propostos por Loureiro (1999).

A engenharia simultânea de sistemas tem se tornado cada vez mais requisitada nas empresas, organizações e instituições, devido à evolução das tecnologias de implementação e ao aumento da complexidade dos próprios sistemas. Logo, novos métodos têm sido desenvolvidos na tentativa de diminuir custos, riscos e tempo gastos nas etapas de modelagem e desenvolvimento do projeto.

Diante deste cenário, os engenheiros de sistemas devem ser capazes de traduzir a necessidade dos *stakeholders* em requisitos do sistema.

Para ter uma modelagem eficaz, que permita ao analista compreender o funcionamento de um sistema, é preciso uma ferramenta computacional, que ofereça as funções necessárias para tal atividade. Apenas com este recurso é possível fazer controles de versões, realizar análises estruturais ou comportamentais do modelo, além de organizar e disponibilizar objetivamente as informações do sistema.

O *software* possibilita também o rastreamento de requisitos, garantindo que durante o processo de engenharia simultânea de sistemas qualquer mudança no projeto seja comunicada aos envolvidos, demonstrando qual a parte do sistema que será afetada com as suas atualizações.

A linguagem *SysML* constitui um grande auxílio no desenvolvimento de produtos complexos, pois apresenta benefícios valiosos para um projeto como: a alteração do modelo de um projeto com menor custo, a facilidade de documentação de um sistema, a reutilização de módulos já desenvolvidos, a rapidez na detecção de erros no início do projeto, garantia do entendimento do sistema por todos os responsáveis do projeto, a redução do tempo gasto nas etapas de concepção e a validação e teste de um sistema.

Uma ressalva a ser feita é que a linguagem *SysML* só se torna completa com a criação de novos estereótipos, o qual possui grande potencial ainda a ser desenvolvido.

Como conclusão, tem-se ainda que a linguagem de *SysML* ainda precisa de adaptações. No entanto, ela já possui os recursos suficientes para uma primeira versão do *template* com base no método proposto por Loureiro (1999), pois esta linguagem permite ao profissional que sejam criados novos tipos de diagramas, já que se trata de uma linguagem não prescritiva.

Esses comentários não se esgotam nesta pesquisa, sendo necessária a continuação da investigação deste novo padrão de modelagem. Portanto, sugere-se a atualização deste *template*, tendo em vista que o LSIS deverá usá-lo em suas atividades de análise sistêmica para atividades do LIT.

Agradecimento

Agradecemos ao CNPq/INPE pela bolsa de Iniciação Científica concedida.

Referências

- FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **A practical guide to SysML**: the systems modeling language. The MK/OMG Press. Elsevier. Amsterdam, 2009.

- LOUREIRO, G. **Engenharia de sistemas**. 2011. São José dos Campos. ITA. Notas de aula do curso de graduação de Engenharia Aeroespacial. ITA, 2011.

- LOUREIRO, G. **A systems engineering and concurrent engineering framework for the integrated development of complex products**. 1991. (PhD Thesis) - Loughborough University, Loughborough, UK. 1999.

ANEXO F - Resumo científico UNITAU 2012: AUXÍLIO COMPUTACIONAL NA APLICAÇÃO DA LINGUAGEM SYSML

AUXÍLIO COMPUTACIONAL NA APLICAÇÃO DA LINGUAGEM SYSML[1]

Maiara Guimarães Flausino[2]
Geilson Loureiro[3]

Resumo

Este trabalho, desenvolvido no Laboratório de Integração e Testes (LIT), do Instituto Nacional de Pesquisas Espaciais (INPE), refere-se à compreensão da importância do emprego de uma ferramenta computacional no processo de modelagem gráfica a partir da engenharia de sistemas, usando a *Systems Modeling Language* (*SysML*) como linguagem padrão descritivo de sistemas complexos. Uma das bases da engenharia de sistemas é a modelagem, principalmente a modelagem gráfica, a qual permite ao profissional ter uma visão do todo, bem como identificar cada relacionamento existente num dado sistema. Evidência disto é a evolução dos *softwares* que dão suporte a engenharia de sistemas, adotados pelo setor produtivo industrial, no sentido de apoiar uma solução balanceada que considera as variáveis como tempo de desenvolvimento, custo dos processos do ciclo de vida, gerenciamento de risco, desempenho do produto e, ainda, que atenda aos requisitos. Para acompanhar o progresso causado pelo aumento da complexidade dos sistemas, os quais necessitam de uma modelagem que considere todas as camadas da sua estrutura de desenvolvimento é fundamental o uso de uma ferramenta computacional que implemente o processo do ciclo de vida do produto ao longo dos processos de engenharia de sistemas, quais sejam: análise de *stakeholders,* análise de requisitos, análise funcional e arquitetura de sistemas. A modelagem de um sistema em uma ferramenta de alto desempenho pode ajudar na rastreabilidade dos requisitos que futuramente serão validados com os *stakeholders*, a fim de auxiliar na construção de modelos. Desde 2007, *softwares* que permitem a modelagem na linguagem *SysML* passaram a estar disponíveis. Para o desenvolvimento desta pesquisa, foram utilizadas três bibliografias específicas sobre o tema em estudo, como também a análise e comparações entre as ferramentas computacionais para a diagramação dos modelos em *SysML*, quais sejam, *Enterprise Architecture (EA)*, *Modelio System Architecture, IBM Rational Rhapsody, Papyrus UML for SysML e TopCased-SysML,* sendo que as duas últimas ferramentas são do tipo *open source*. Neste trabalho realizou-se pesquisa sobre a linguagem *SysML* e suas ferramentas, esta linguagem pode ser usada para modelar produtos e organizações, pois permite ao profissional que sejam criados novos tipos de diagramas, já que se trata de uma linguagem não prescritiva. Para colocar em prática o aprendizado da linguagem *SysML*, foi construída a primeira versão de um *template* dos modelos, de engenharia de sistemas ao longo do seu processo, de um produto espacial. O *software* utilizado para modelar o *template* foi o *IBM Rational Rhapsody,* por ser uma ferramenta com diversas funções, entre elas rastreabilidade de requisitos, interface gráfica amigável, integração com o *IBM Rational Doors, software* que auxilia o processo de engenharia de requisitos, geração de códigos e relatórios automaticamente.

Palavras-chave: Linguagem *SysML*, *Softwares*, Engenharia de Sistemas.

---

[1]XVII Encontro de Iniciação Científica

[2]Graduação, Universidade Federal de São Carlos - UFSCar, maiara.flausino@lit.inpe.br
[3]Pós-Doutorado, Instituto Nacional de Pesquisas Espaciais - INPE, geilson@lit.inpe.br

ANEXO G – Pôster SICINPE 2012: SYSML PARA ENGENHARIA SIMULTANEA DE SISTEMAS

SICINPE - 2012
Seminário de Iniciação Científica e Iniciação em Desenvolvimento Tecnológico e Inovação

SYSML PARA ENGENHARIA SIMULTÂNEA DE SISTEMAS ESPACIAIS

Maiara Guimarães Flausino
(UFSCar, Bolsista PIBIC/CNPq)
maiara.flausino@lit.inpe.br

Geilson Loureiro
(LIT/INPE, Orientador)
geilson@lit.inpe.br

1. INTRODUÇÃO

A Engenharia Simultânea de Sistemas requer uma visão de todas as fases do ciclo de vida de um produto. Evidência disto é a evolução das técnicas de modelagem de sistemas complexos, uma vez que os modelos permitem ao profissional ter uma visão do todo, bem como identificam cada relacionamento existente em um dado sistema.

2. OBJETIVO

Diante do exposto, este trabalho tem como objetivo contribuir para o entendimento do método de Engenharia Simultânea de Sistemas, proposto por Loureiro (1999), desenvolvendo um template para aplicar o método, usando a Systems Modeling Language (SysML), linguagem de modelagem descritiva de sistemas complexos.

3. METODOLOGIA

Com base na análise crítica da literatura científica pesquisada, foi feita a extração dos conceitos teóricos sobre a linguagem SysML e a Engenharia Simultânea de Sistemas. Esses conceitos ofereceram os fundamentos a partir dos quais o sistema CANSAT foi modelado. Para colocar em prática o aprendizado deste projeto, foi necessário o treinamento da ferramenta IBM Rhapsody.

4. DESENVOLVIMENTO

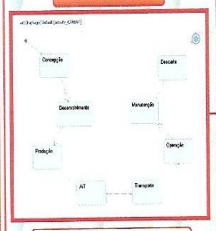
Este projeto buscou modelar na linguagem SysML um produto espacial (Sistema CANSAT) seguindo o método proposto por Loureiro (1999). Como este foi modelado primeiramente em Análise Estruturada, foi preciso um estudo constante para adaptar e criar novos tipos de diagramas e relacionamentos. Adicionalmente, foram realizadas discussões técnicas com o orientador sobre o assunto.

5. RESULTADOS

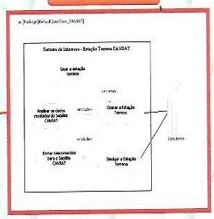
Diagramas da linguagem SysML modelados para o template deste trabalho

Comportamento

Atividades

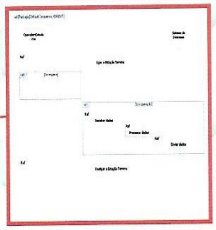


Caso de Uso

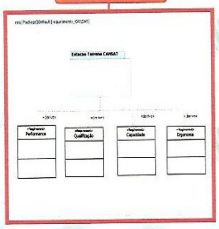


Máquina de Estados

Sequência

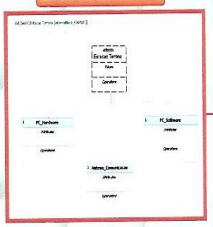


Requisitos

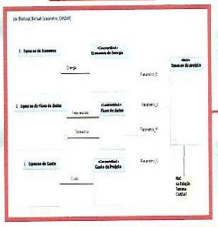


Estrutura

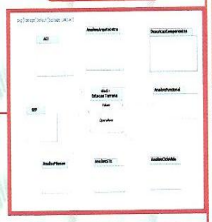
Blocos Internos



Parâmetros



Pacotes

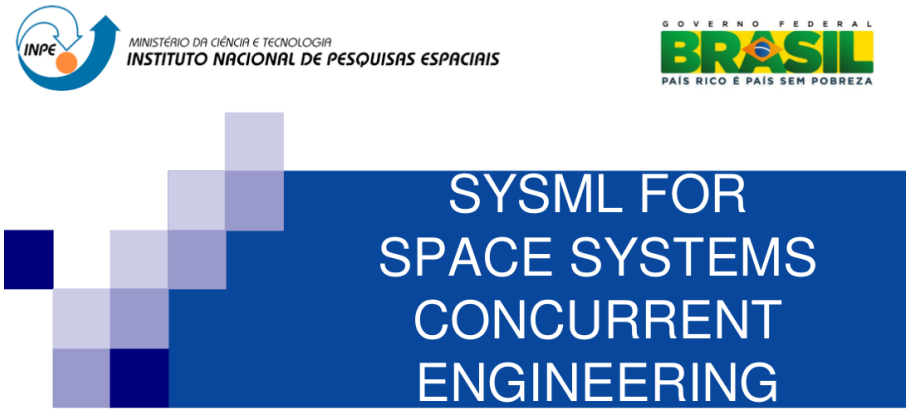


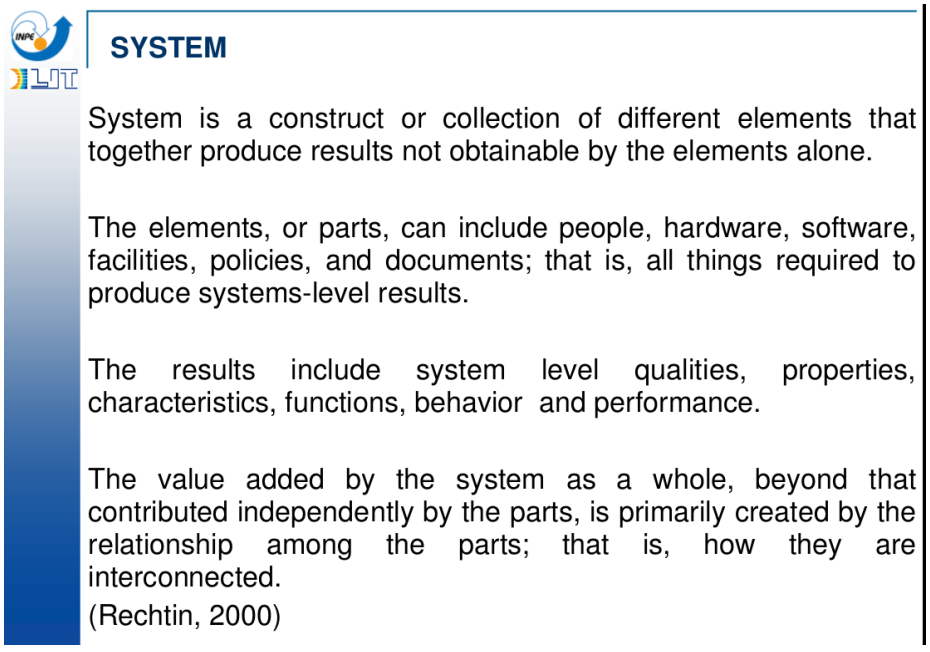
Definição de Blocos

6. CONSIDERAÇÕES FINAIS

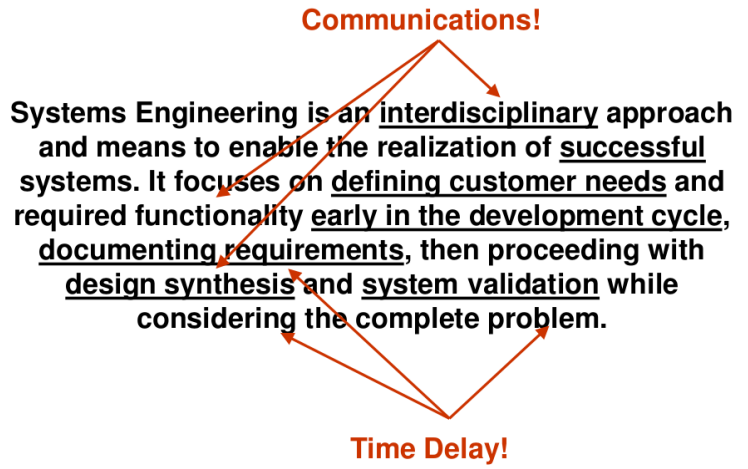
Conclui-se que a linguagem SysML ainda precisa de adaptações. No entanto, ela possui recursos suficientes para uma implementação do método desenvolvido por Loureiro (1999), já que permite a elaboração de novos tipos de diagramas e relacionamentos, pois se trata de uma linguagem não prescritiva. Desta forma, para obter uma modelagem consistente, é preciso um software de alto desempenho, tendo em vista que somente com esse recurso computacional é possível realizar análises estruturais ou comportamentais de um modelo, possibilitando o rastreamento de requisitos.
A seguir são apresentados exemplos de projetos realizados no INPE relacionados à Engenharia de Sistemas, nos quais a linguagem SysML contribuiria de forma positiva: a) Projeto SACI (Satélite Avançado de Comunicações Interdisciplinares). b) LSIS (Laboratório de Engenharia de Sistemas). c) ENGESIS (Engenharia, Gestão e Simulação de Sistemas Espaciais).

ANEXO H - Apresentação LSIS 2011: SYSML PARA ENGENHARIA SIMULTANEA DE SISTEMAS





SYSTEMS ENGINEERING

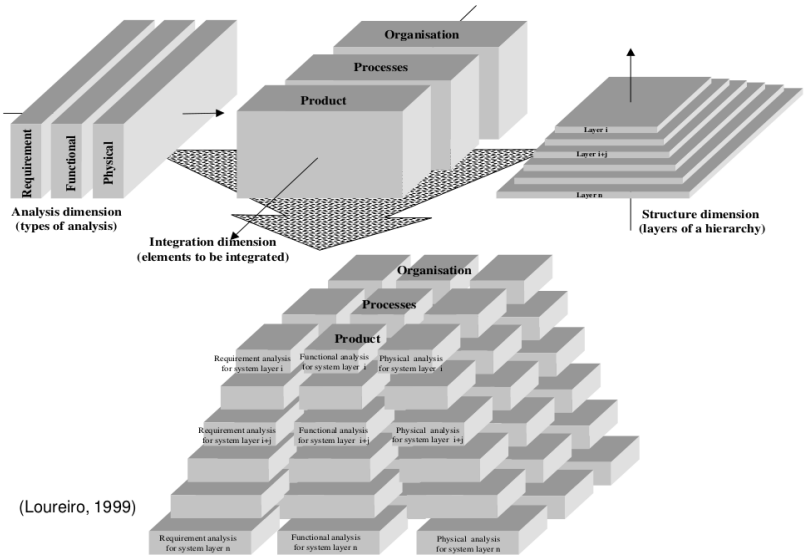


PRINCIPLES OF SYSTEMS ENGINEERING

- The whole is the system life
- The system solution is composed of product and organization elements
- Systems exist in a hierarchy
- Systems are architected top down and are AIT bottom up
- Systems are architected outside in and are AIT inside out
- Separate the problem domain from the solution domain
- Systems are architected alternating between the problem (essential) and the solution (implementation) domain
- Systems are architected alternating between 4 hierarchies: stakeholders, requirements, functions, implementation
- Architectures are fully described by structure and behavior
- Architectures seek to minimize coupling and maximize cohesion
- Start from normal behavior then derive exception handling

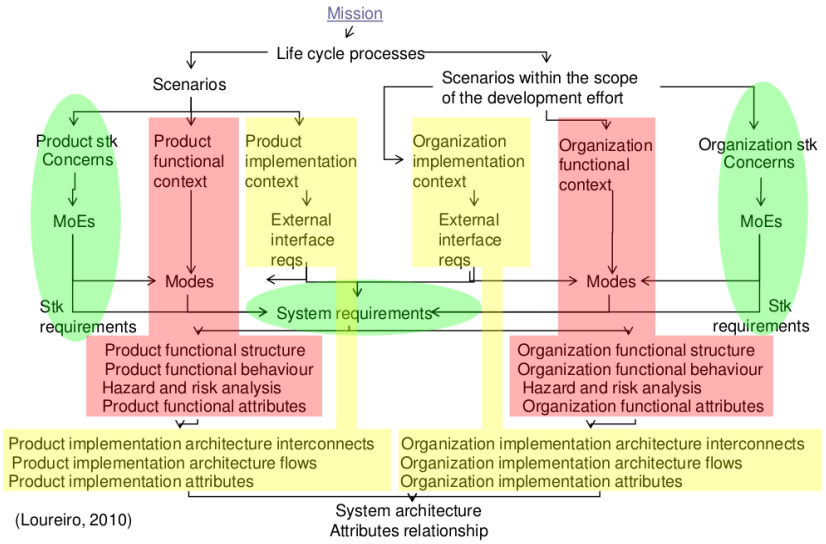
(Loureiro, 2011)

FRAMEWORK FOR INTEGRATED DEVELOPMENT



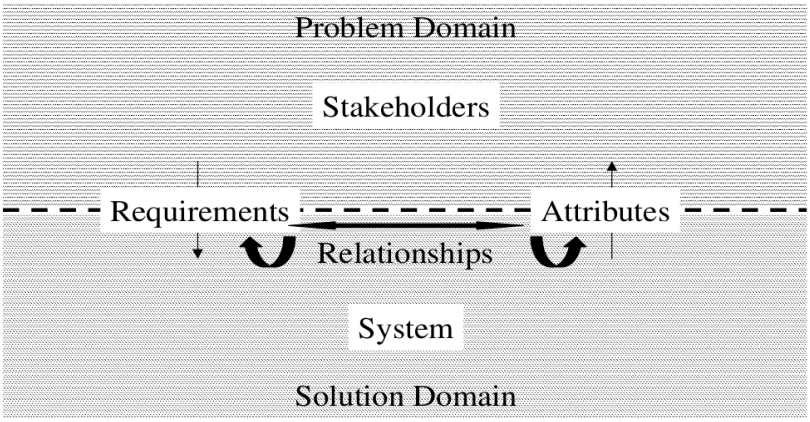
(Loureiro, 1999)

METHOD OVERVIEW



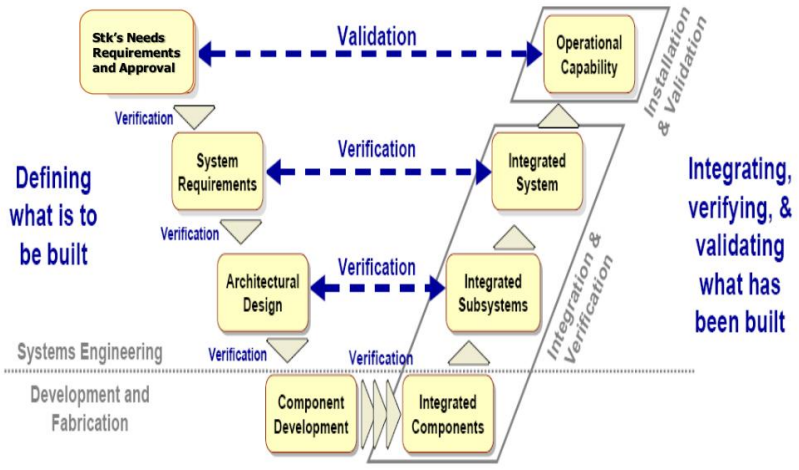
(Loureiro, 2010)

PROBLEM AND SOLUTION DOMAINS



(Loureiro, 1999)

THE V MODEL



(Stevens et al., 1998)

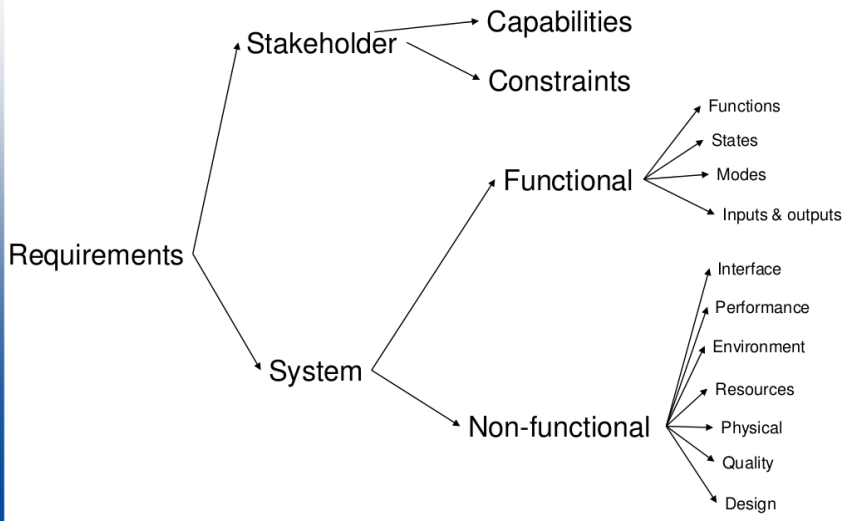
THE V MODEL - CONT.

The Simple Life Cycle can be reorganized as a V-diagram to emphasize:

- Verification between phases, checking what has been built against its requirements.
- Validation as end-to-end verification ensuring that the complete system meets the users needs.
- Decomposition and definition of what is to be built
- Integrating and verifying what has been built.

(Stevens et al, 1998)

REQUIREMENTS



STAKEHOLDERS

Individuals and organizations actively involved in the project, or whose interests may be affected as a result of project execution or project completion. They may also exert influence over the projects objectives and outcomes.

(PMBOK, 2004)

People or organizations;

Affect product or are affected by product;

Affect product life cycle organization or are affected by product life cycle organization;

Or even people who think they affect or are affected.

(Loureiro, 2011)

REQUIREMENT

Requirement is a necessary attribute in a system a statement that identifies a capability, characteristic, or quality factor of a system in order for it to have value and utility to a customer or user.

(Young, 2004)

A statement that defines:

- What the stakeholders, in a potential new system, need
- What the system must do in order to satisfy that need

(Hull et al, 2005)

The term requirements refers to the total set of considerations that govern: what is to be accomplished, how well it is to be accomplished, and under what conditions it is to be accomplished. They also govern, as appropriate, logical, and physical characteristics of the system.

(EIA-632)

ARCHITECTURE

Functional architecture consists of:

- The functional context of the system
- The structure and behavior of the system function
- The structure and behavior of the functions obtained from the stepwise decomposition of the system function
- The hierarchy of functions resulting from the stepwise decomposition

Physical architecture consists of:

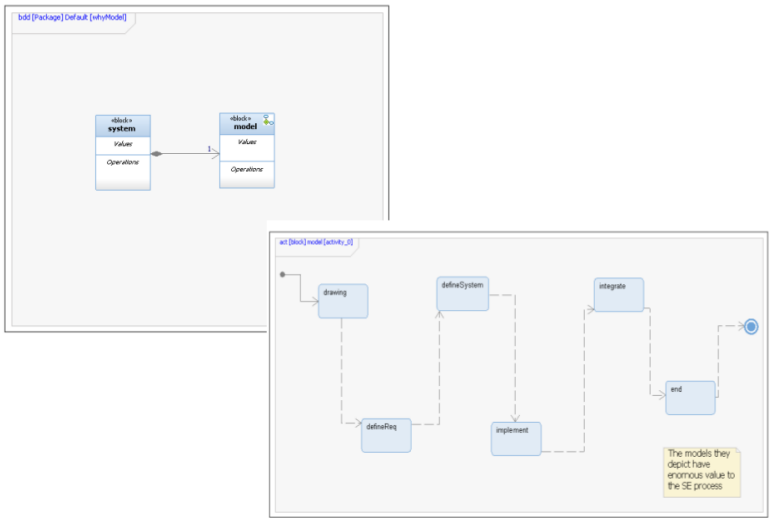
- The physical context of the system
- The structure and behavior of the system
- The structure and behavior of the physical elements obtained from the stepwise decomposition of the system
- The hierarchy of physical elements resulting from the stepwise decomposition

WHY MODELING IS IMPORTANT?

Modeling helps to:

- improve design quality, reduce errors and avoid ambiguity.
- improve communications.
- manage complex systems development.
- separate different concerns.
- hierarchical modeling.
- facilitates impact analysis of requirements and design changes.
- supports incremental development.

WHY MODEL?



UML – UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

Development of UML began in late 1994 when Grady Booch and Jim Rumbaugh of Rational Software Corporation began their work on unifying the Booch and OMT (Object Modeling Technique) methods.

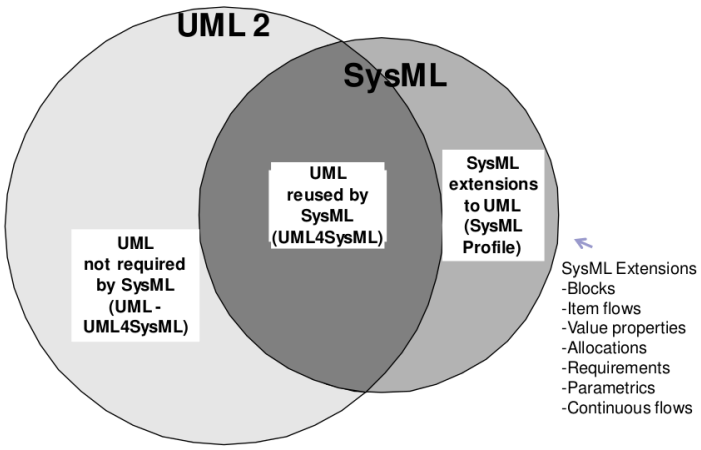
In the Fall of 1995, Ivar Jacobson and his Objectory company joined Rational and this unification effort, merging in the OOSE (Object-Oriented Software Engineering) method.

As the primary authors of the Booch, OMT, and OOSE methods, Grady Booch, Jim Rumbaugh, and Ivar Jacobson were motivated to create a unified modeling language as all were working on common things.

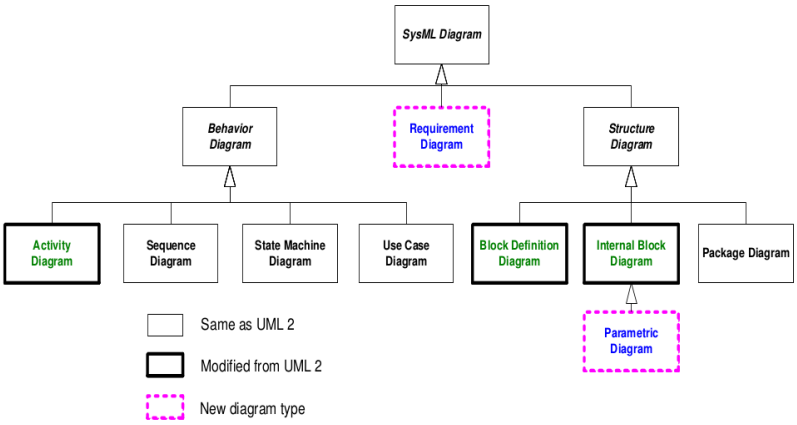
SYSML - Systems Modeling Language

- It is a visual modeling language that provides:
  - Semantics = meaning
  - Notation = representation of meaning
- It is not a methodology or a tool:
  - SysML is methodology and tool independent

RELATIONSHIP BETWEEN SYSML AND UML



SYSML DIAGRAM TAXONOMY



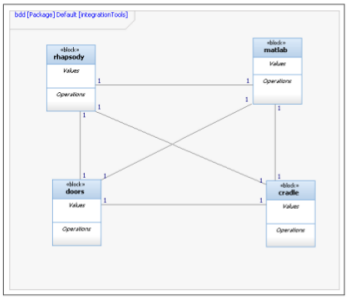
SYSML DRAWING

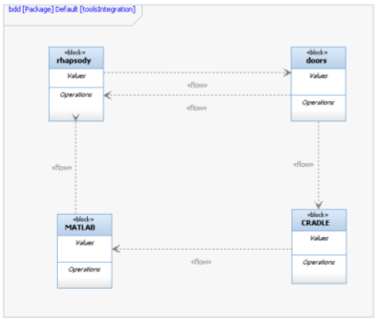
SYSML DIAGRAM	PURPOSE	UML ANALOG
Activity diagram	Show system behavior as control and data flows. Useful for functional analysis. Compare Extended Functional Flow Block diagrams (EFFBDs), already commonly used among systems engineers.	Activity diagram
Block Definition diagram	Show system structure as components along with their properties, operations and relationships. Useful for system analysis and design.	Class diagram
Internal Block diagram	Show the internal structures of components, including their parts and connectors. Useful for system analysis and design.	Composite Structure diagram
Package diagram	Show how a model is organized into packages, views and viewpoints. Useful for model management.	Package diagram
Parametric diagram	Show parametric constraints between structural elements. Useful for performance and quantitative analysis.	N/A
Requirement diagram	Show system requirements and their relationships with other elements. Useful for requirements engineering.	N/A

SYSML DRAWING - CONT.

Sequence diagram	Show system behavior as interactions between system components. Useful for system analysis and design.	Sequence diagram
State Machine diagram	Show system behavior as sequences of states that a component or interaction experience in response to events. Useful for system design and simulation/code generation.	State Machine diagram
Use Case diagram	Show system functional requirements as transactions that are meaningful to system users. Useful for specifying functional requirements. (Note potential overlap with Requirement diagrams.)	Use Case diagram
Allocation tables* *dynamically derived tables, not really a diagram type	Show various kinds of allocations (e.g., requirement allocation, functional allocation, structural allocation). Useful for facilitating automated verification and validation (V&V) and gap analysis.	N/A

TOOLS INTEGRATIONS

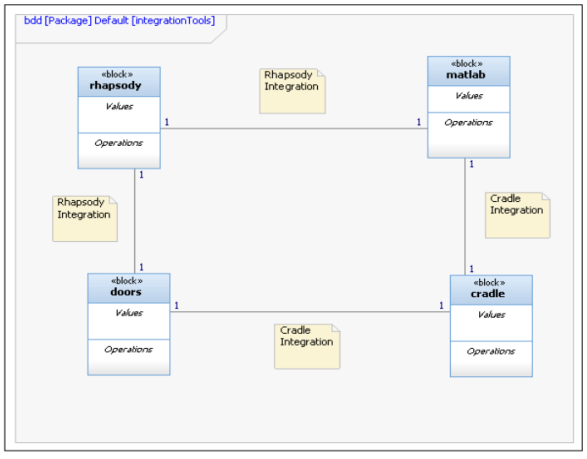




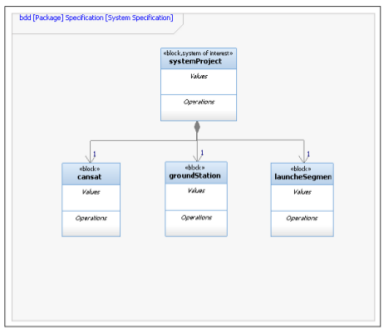
TOOLS INTEGRATION - PROBLEMS

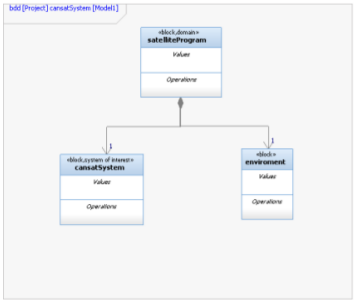
- Each tool / application has different repository and uses different data structures to store information.
- The tools are closed code, not providing direct access to data.

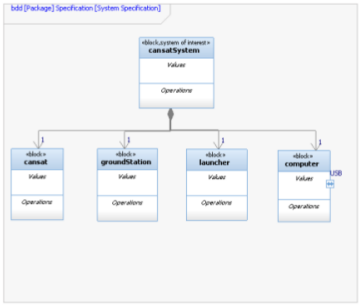
INTEGRATION - OVERVIEW

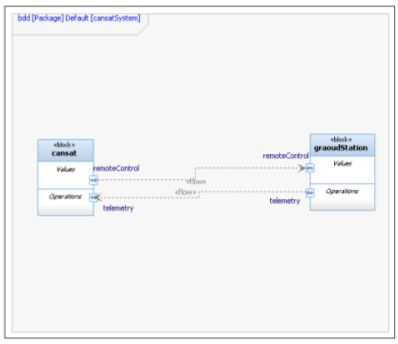


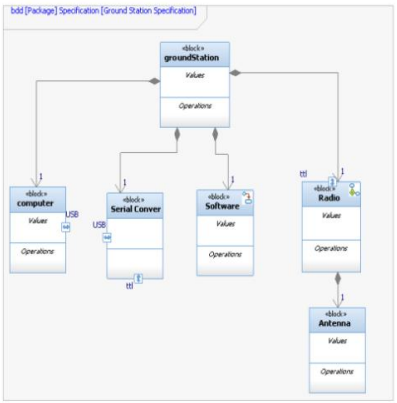
CANSAT SYSTEM

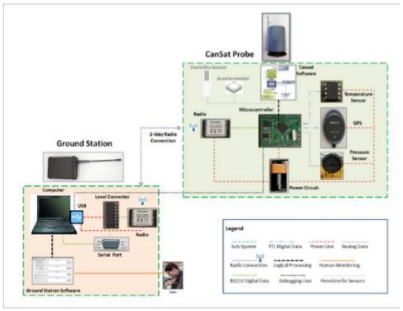


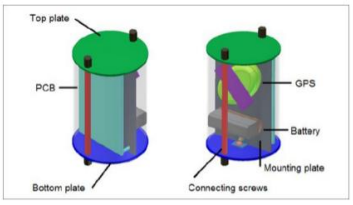


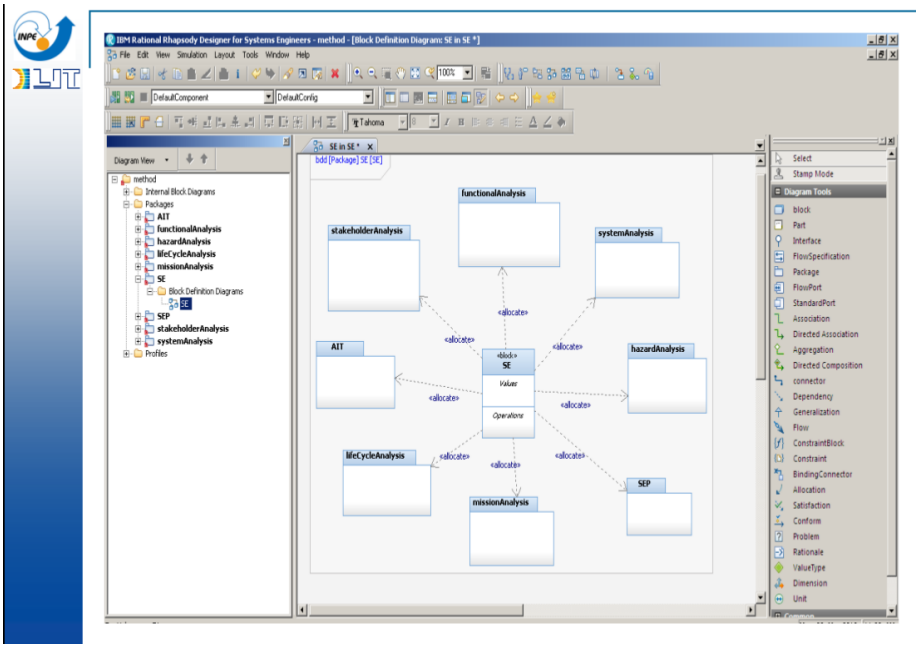

bdd [Package] Specification [System Specification]
«block,system of interest»
cansatSystem
Values
Operations
«block»
canSat
«block»
groundStation
«block»
launcher
«block»
computer
USB


bdd [Package] Default [cansatSystem]
«block»
cansat
Values
Operations
remoteControl
telemetry
«flow»
«block»
graoudStation
Values
Operations

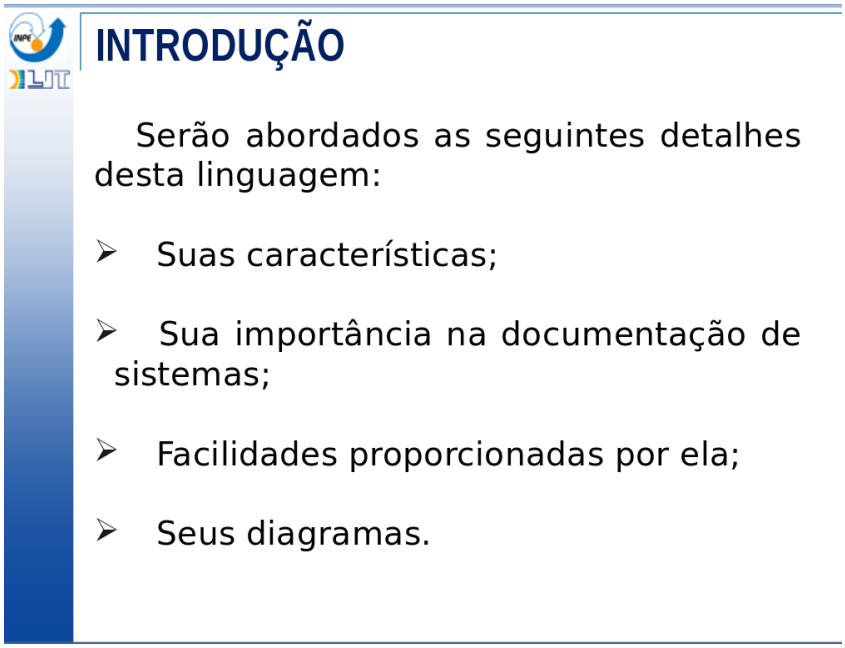

bdd [Package] Specification [Ground Station Specification]
«block»
groundStation
Values
Operations
«block»
computer
USB
«block»
Serial Conver
USB
«block»
Software
Values
Operations
«block»
Radio
Values
Operations
«block»
Antenna
Values
Operations


CanSat Probe
Ground Station
2-Way Radio Connection
Computer
Level Converter
USB
Serial Port
Radio
Ground Station Software
Legend


Top plate
PCB
Bottom plate
GPS
Battery
Mounting plate
Connecting screws

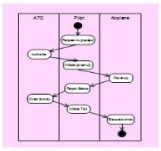

IBM Rational Rhapsody Designer for Systems Engineers - method - [Block Definition Diagram: SE in SE *]
File Edit View Simulation Layout Tools Window Help
Diagram View
method
Internal Block Diagrams
Packages
AIT
functionalAnalysis
hazardAnalysis
lifeCycleAnalysis
missionAnalysis
SE
Block Definition Diagrams
SE
SEP
stakeholderAnalysis
systemAnalysis
Profiles
SE in SE *
bdd [Package] SE [SE]
functionalAnalysis
stakeholderAnalysis
systemAnalysis
AIT
hazardAnalysis
lifeCycleAnalysis
missionAnalysis
SEP
«allocate»
«block»
SE
Value
Operations
Select
Stamp Mode
Diagram Tools
block
Part
Interface
FlowSpecification
Package
FlowPort
StandardPort
Association
Directed Association
Aggregation
Directed Composition
connector
Dependency
Generalization
Flow
ConstraintBlock
Constraint
BindingConnector
Allocation
Satisfaction
Conform
Problem
Rationale
ValueType
Dimension
Unit

ANEXO I - Apresentação LSIS 2012: A Linguagem SysML



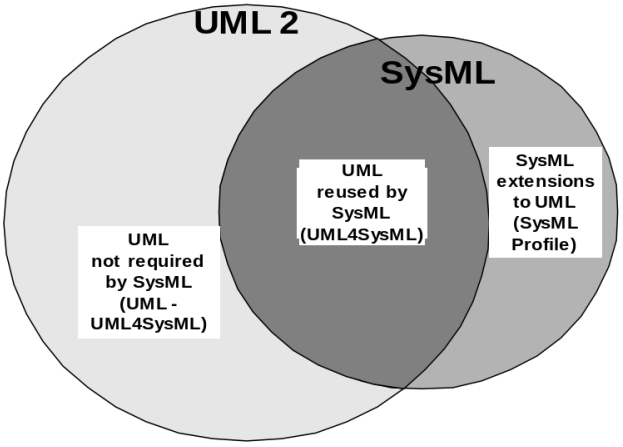
SYSML - SYSTEMS MODELING LANGUAGE

Trata-se de uma linguagem de modelagem visual que possui uma sintaxe e uma semântica muito bem definidas, garantindo que não haja ambiguidade na construção de modelos de sistemas complexos.



Fonte: FRIEDENTHAL,2009.

RELAÇÃO ENTRE SYSML E UML

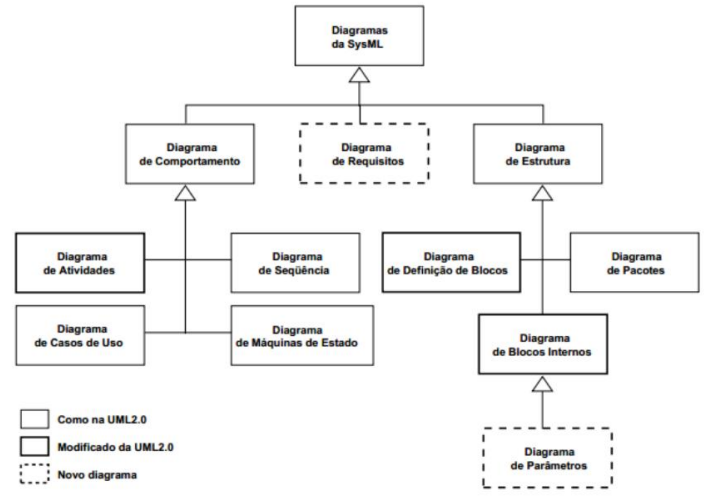


Fonte: FRIEDENTHAL,2009.

UML - UNIFIED MODELING LANGUAGE

Trata - se de uma linguagem gráfica considerada padrão para especificar, visualizar, construir e documentar somente softwares.

DIAGRAMAS DA LINGUAGEM SYSML

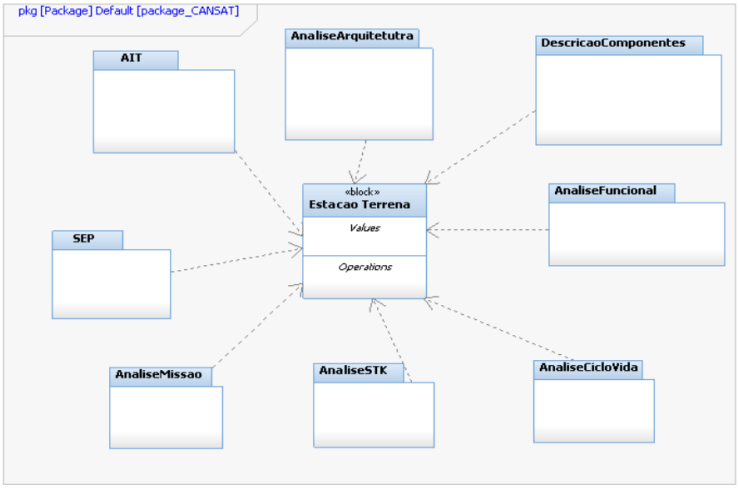


Fonte: FLAUSINO, 2012.

DIAGRAMA DE PACOTES

- Mostra como um modelo está organizado em pacotes, opiniões e diferentes pontos de vista.
- Útil para a gestão do modelo.
- Usado especialmente para ilustrar a arquitetura de um sistema.

DIAGRAMA DE PACOTES (CONT.)

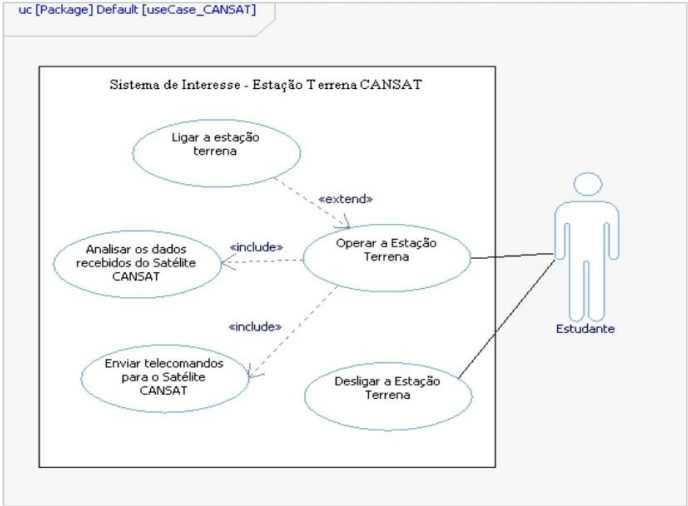


Fonte: FLAUSINO, 2012.

DIAGRAMA DE CASO DE USO

- Usado para validar os requisitos de um sistema com os próprios *stakeholders.*
- Descreve as funcionalidades propostas para um novo sistema.

DIAGRAMA DE CASO DE USO (CONT.)

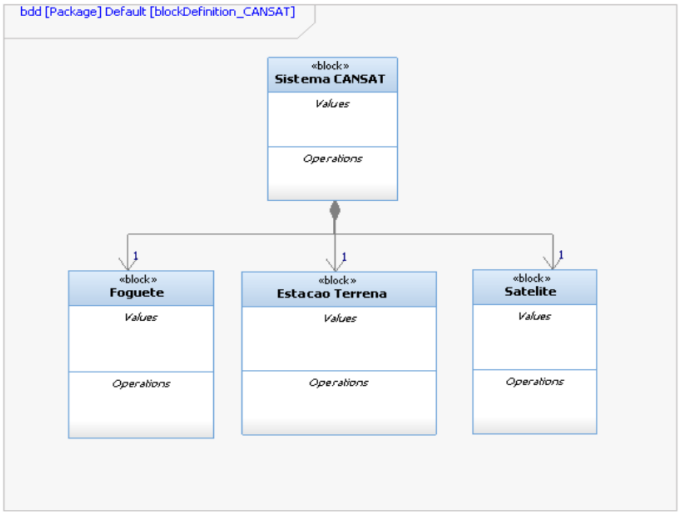


Fonte: FLAUSINO, 2012.

DIAGRAMA DE DEFINIÇÃO DE BLOCOS

- Mostra a estrutura de um sistema com suas respectivas propriedades e operações.
- Descreve as relações entre cada subsistema com o sistema e as trocas de fluxo de material, energia e informação.

DIAGRAMA DE DEFINIÇÃO DE BLOCOS (CONT.)

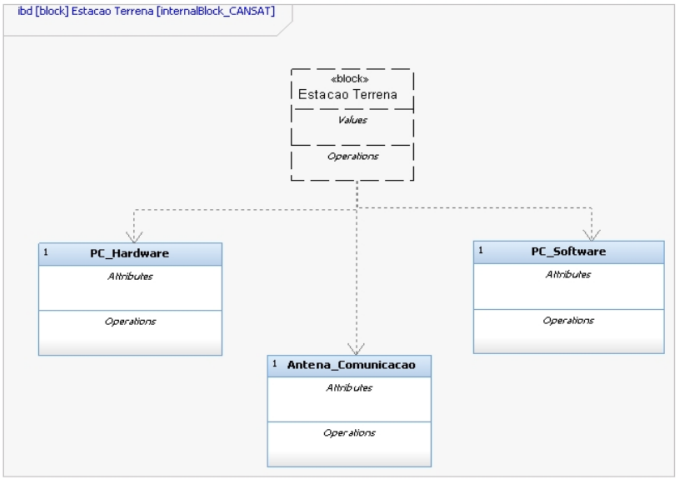


Fonte: FLAUSINO, 2012.

DIAGRAMA DE BLOCOS INTERNOS

- Útil para análise em detalhes de um sistema.
- Descreve as relações entre cada subsistema e a troca de fluxo (material, energia e informação) entre eles.
- Modela a visão estática do projeto, mostrando os conjuntos das partes e seus relacionamentos, no nível mais baixo de abstração.

DIAGRAMA DE BLOCOS INTERNOS (CONT.)



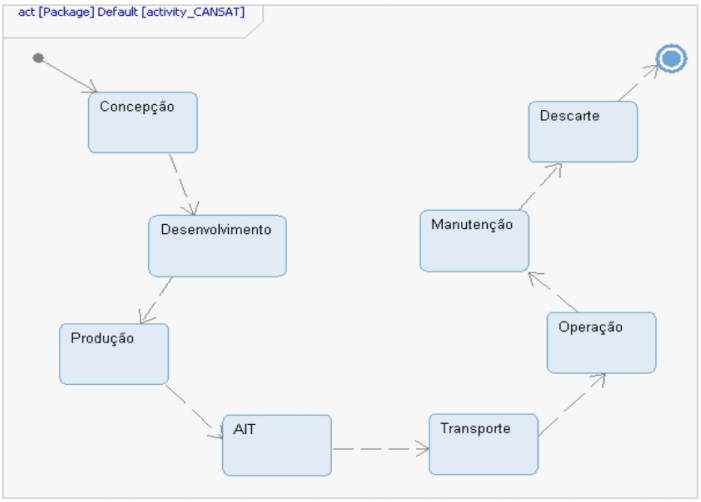
Fonte: FLAUSINO, 2012.

DIAGRAMA DE ATIVIDADES

- Exibe etapas de um processo, com foco no fluxo de material, energia e informação entre uma etapa e outra do sistema.

- Existem dois modos de criar o diagrama de atividades:
  - Fluxo de trabalho: especifica, constrói e documenta processos.
  - Operação: empregado como fluxograma, pois contém ramificações, bifurcações e estados de união.

DIAGRAMA DE ATIVIDADES (CONT.)

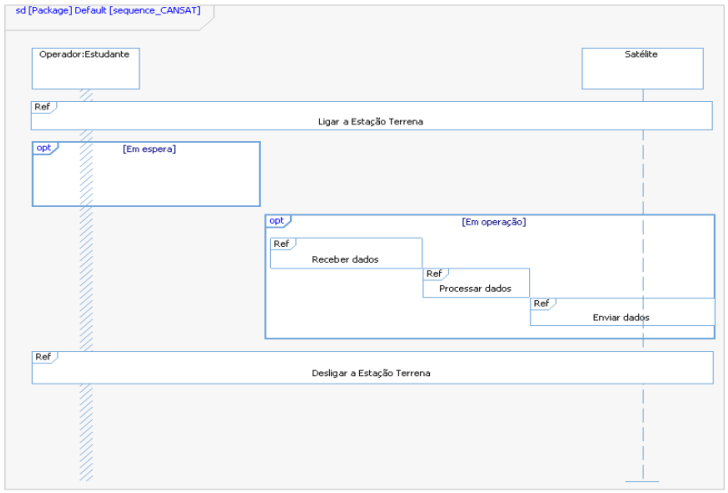


Fonte: FLAUSINO, 2012.

DIAGRAMA DE SEQUÊNCIA

- Descreve de forma simples e lógica como as atividades se comportam ao longo do tempo em um sistema.
- Enfatiza a ordenação temporal das atividades de um sistema, através das mensagens que são trocadas entre os eventos do sistema.

DIAGRAMA DE SEQUÊNCIA (CONT.)

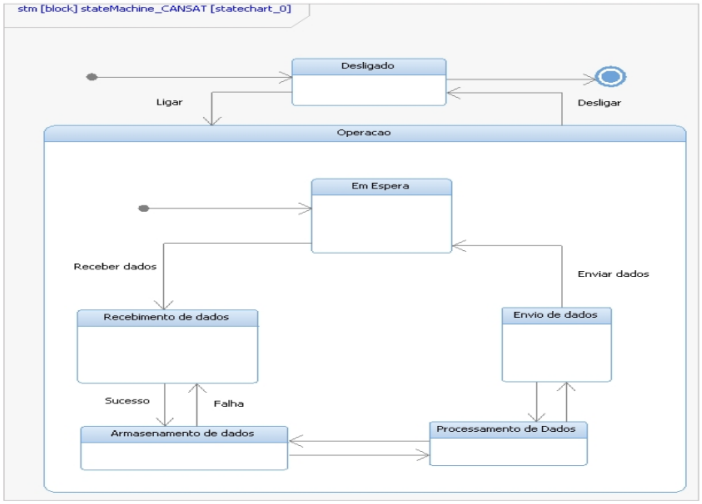


Fonte: FLAUSINO, 2012.

DIAGRAMA DE MÁQUINA DE ESTADOS

- Demonstra a visão dinâmica de um sistema.
- É formado por estados, transições de eventos e de atividades.
- Há softwares de modelagem que permitem animações e simulações no próprio modelo, podendo também gerar códigos automaticamente.

DIAGRAMA DE MÁQUINA DE ESTADOS (CONT.)

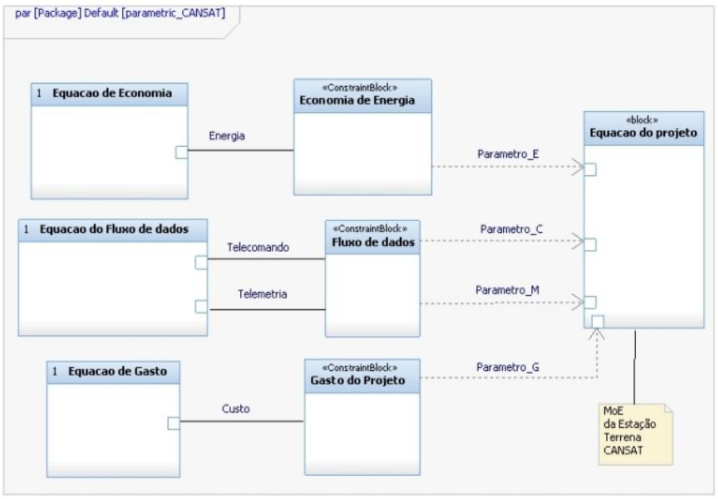


Fonte: FLAUSINO, 2012.

DIAGRAMA PARAMÉTRICO

- Derivam-se a partir dos parâmetros e das restrições de um sistema.
- Descreve sucintamente as variáveis do processo, as quais, sendo cumpridas, habilitam a satisfação dos requisitos.

DIAGRAMA PARAMÉTRICO (CONT.)

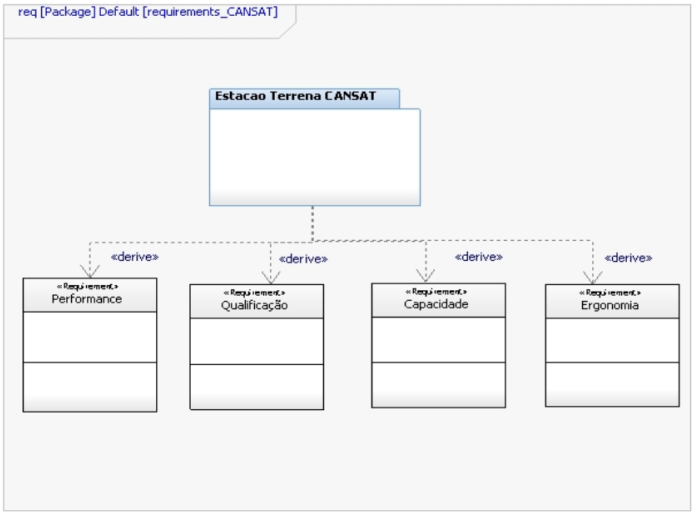


Fonte: FLAUSINO, 2012.

DIAGRAMA DE REQUISITOS

- Trata-se de um tipo específico do Diagrama de Definição de Blocos.
- Usado para modelar tanto os requisitos de *stakeholder* como os requisitos do sistema, mostrando os seus relacionamentos.

DIAGRAMA DE REQUISITOS (CONT.)



Fonte: FLAUSINO, 2012.

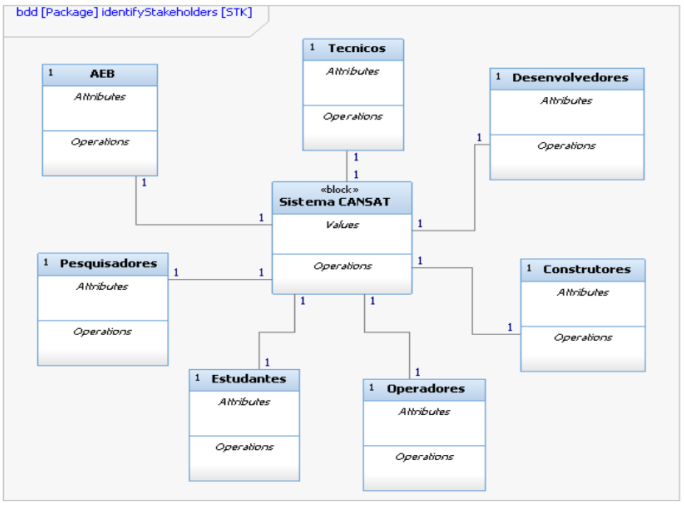
DIAGRAMA DE CONTEXTO

O Diagrama de Contexto pode ser usado também para representar tanto o contexto funcional como o contexto físico de um produto.

As principais características deste novo estereótipo de diagrama:

- Apresenta um conjunto de elementos relacionados para um determinado propósito, como especificar um *stakeholder*.
- Define o ambiente em que o sistema vai pertencer, demonstrando as suas características.
- É composto por fluxos de material, energia e informação, que mostram as interfaces entre o sistema e as entidades externas.
- Permite identificar os limites dos processos, as áreas envolvidas com o processo e os relacionamentos com outros

DIAGRAMA DE CONTEXTO (CONT.)



Fonte: FLAUSINO, 2012.

CONSIDERAÇÕES FINAIS

Para modelar um sistema, há mais de uma dimensão a ser considerada, ou seja, o modelador deve conhecer em detalhes:

- a linguagem que será usada na modelagem;
- a ferramenta na qual a modelagem será realizada;
- o sistema que será modelado.

Sem isso, não há "empregabilidade" desta linguagem, pois ela perde a sua funcionalidade, que é representar o ambiente de forma fiel.

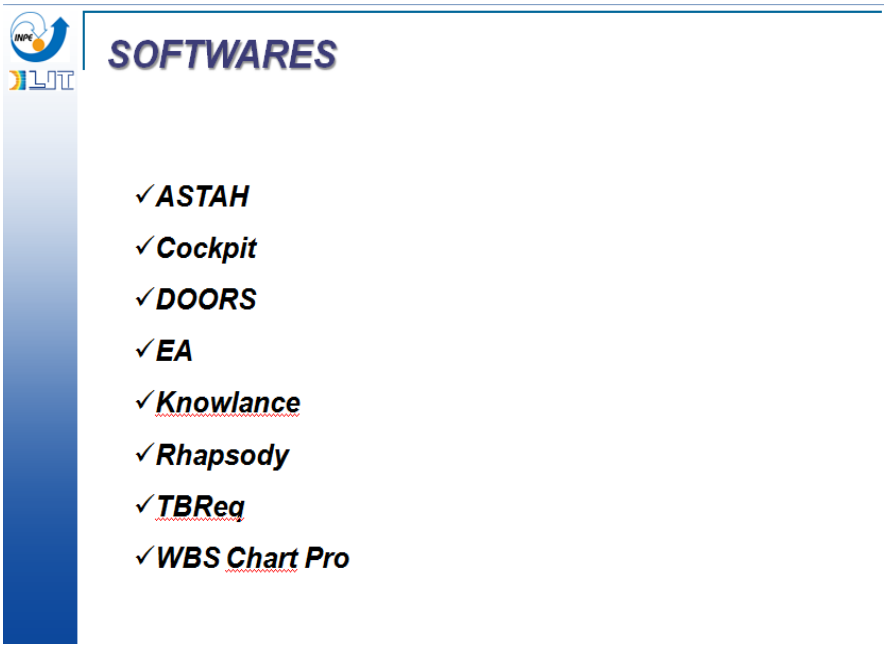
BIBLIOGRAFIA

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **A practical guide to SysML:** the systems modeling language. The MK/OMG Press. Elsevier. Amsterdam, 2009.

HOLT, J.; PERRY, S. **SysML for systems engineering.** IET. Stevenage. 2008.

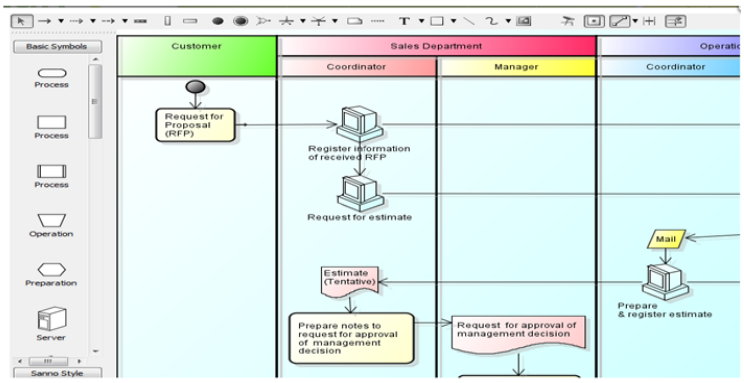
WEILKIENS, T. **Systems engineering with SysML/UML**: modeling, analysis, design. The MK/OMG Press. Elsevier. Amsterdam, 2009.

ANEXO J - Apresentação LSIS 2013: Análise de softwares para o LSIS



ASTAH

For years, Astah has been a leader in UML modeling platforms. We are utilizing that same spirit of innovation and intuitive design as we develop our newest product: Astah SysML.

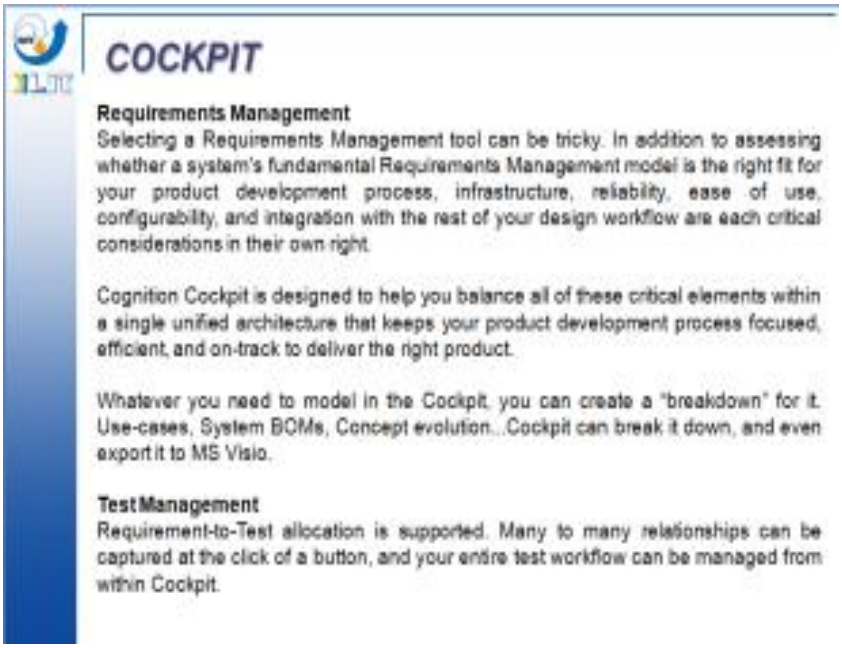


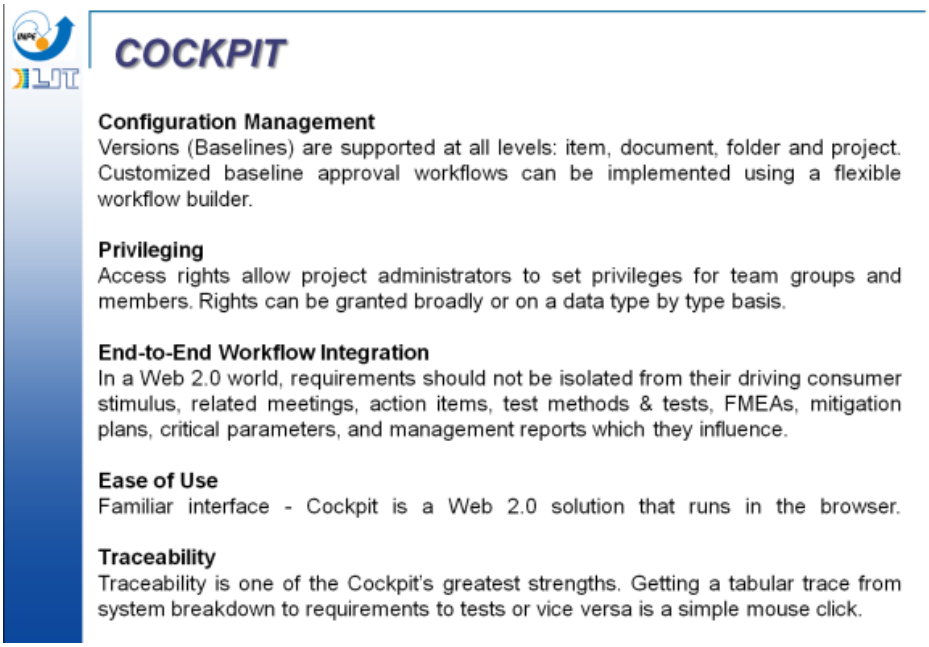
ASTAH

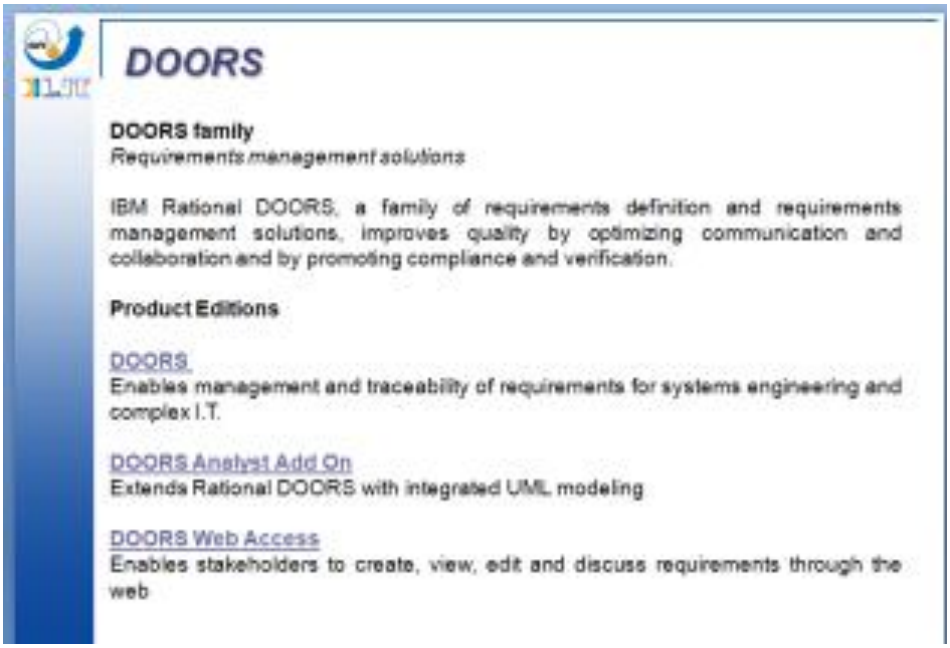
Following the guidelines for SysML laid out by OMG, Astah SysML represents a broad use platform perfect a wide variety of modeling tasks such as; specifying, analyzing, designing, verifying any number of complex systems.

These models are applicable in an even wider variety of systems driven fields including; hardware, procedures, facilities, personal information, software, and, of course, many more.

Astah SysML has entered its beta testing phase and we need your feedback to make Astah SysML perfectly suited for our users.


COCKPIT
Requirements Management
Selecting a Requirements Management tool can be tricky. In addition to assessing whether a system's fundamental Requirements Management model is the right fit for your product development process, infrastructure, reliability, ease of use, configurability, and integration with the rest of your design workflow are each critical considerations in their own right.
Cognition Cockpit is designed to help you balance all of these critical elements within a single unified architecture that keeps your product development process focused, efficient, and on-track to deliver the right product.
Whatever you need to model in the Cockpit, you can create a "breakdown" for it. Use-cases, System BOMs, Concept evolution...Cockpit can break it down, and even export it to MS Visio.
Test Management
Requirement-to-Test allocation is supported. Many to many relationships can be captured at the click of a button, and your entire test workflow can be managed from within Cockpit.


COCKPIT
Configuration Management
Versions (Baselines) are supported at all levels: item, document, folder and project. Customized baseline approval workflows can be implemented using a flexible workflow builder.
Privileging
Access rights allow project administrators to set privileges for team groups and members. Rights can be granted broadly or on a data type by type basis.
End-to-End Workflow Integration
In a Web 2.0 world, requirements should not be isolated from their driving consumer stimulus, related meetings, action items, test methods & tests, FMEAs, mitigation plans, critical parameters, and management reports which they influence.
Ease of Use
Familiar interface - Cockpit is a Web 2.0 solution that runs in the browser.
Traceability
Traceability is one of the Cockpit's greatest strengths. Getting a tabular trace from system breakdown to requirements to tests or vice versa is a simple mouse click.


DOORS
DOORS family
Requirements management solutions
IBM Rational DOORS, a family of requirements definition and requirements management solutions, improves quality by optimizing communication and collaboration and by promoting compliance and verification.
Product Editions
DOORS
Enables management and traceability of requirements for systems engineering and complex I.T.
DOORS Analyst Add On
Extends Rational DOORS with integrated UML modeling
DOORS Web Access
Enables stakeholders to create, view, edit and discuss requirements through the web


Enterprise Architect (EA)
Enterprise Architect (EA) representa um divisor de águas em relação às ferramentas de modelagem de processos, dados e sistemas, sendo ao mesmo tempo: robusta, de fácil utilização/aprendizado e que oferece o melhor custo-benefício do mercado: Suporte total ao ciclo de vida de modelagem de processos, dados e sistemas.
Contempla Inúmeras notações e técnicas (BPMN, UML, Modelagem de Dados, SOAML, SysML, ARCGis e outras).
Permite a utilização de técnicas de levantamento e documentação de Requisitos
Abordagem completa em Análise e Projeto de Sistemas conforme a UML

Enterprise Architect (EA)

Automatização na engenharia de código (geração, reversa e sincronização), contemplando múltiplas linguagens de programação

Geração de Documentação de apoio e Relatórios personalizados (em HTML e RTF)
Rastreabilidade entre todos os elementos de modelagem (processos, regras, requisitos, casos de uso, classes, componentes, tabelas, etc).

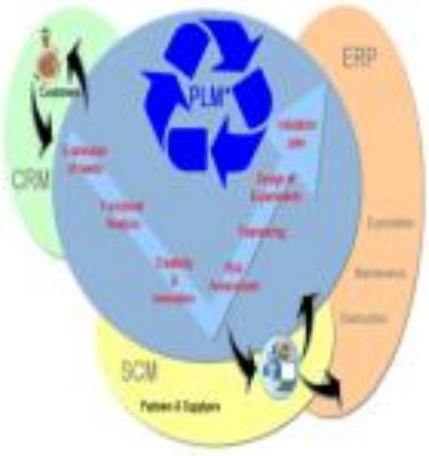
Integração com ferramentas de gerenciamento de configuração e versionamento (como CVS, Subversion, SourceSafe, entre outras).

Workgroup, possibilitando uso compartilhado e seguro pelos usuários.

Exportação/troca de informações com outras ferramentas via XML.

Permite criar protótipos de telas para validação e rastreabilidade de documentação.

Knowllence



- TDC Structure: Software for internal functional analysis
- Functional Bloc Diagramm - SADT - functional analysis charts.
- Crossing functions and components, needs and solutions.
- TDC System: Collaborative platform for system engineering and project management

Knowllence

A ferramenta gráfica Bloco Diagrama Funcional (BDF) permite identificar as funções técnicas internas (Fluxos fechados) induzidas pela concepção. Estas funções serão igualmente tratadas, depois, no TDC FMEA se continuar por uma AMFEC.

A coerência dos dados metodológicos é assegurada pela partilha dos dados entre as diferentes ferramentas ao longo do processo de concepção: análise funcional externa (CdCF) e interna (BDF-QAF), AMFEC Produto, AMFEC Processo, Plano de vigilância, Diagrama de fluxo, Características especiais, etc.

São considerados todos os dados (soluções, componentes, funções técnicas, etc.). Como conhece todas as datas de alteração, os autores e os conteúdos, o utilizador pode, por exemplo, filtrar os dados que foram alterados desde a última revisão do projeto ou ainda listar os sucessivos desenvolvimentos de uma característica componente.

A utilização em projeção mural durante uma sessão de trabalho facilita a reflexão do grupo de trabalho com ferramentas tais como a arborescência das componentes (nomenclatura), o BDF e o QAF. A realização e, sobretudo, a alteração em direto permitem ganhar em reatividade.

Rhapsody

A collaborative, model-based systems engineering development environment

IBM® Rational® Rhapsody® Designer for Systems Engineers is a model-based systems engineering (MBSE) environment using the industry-standard Systems Modeling Language (SysML) and Unified Modeling Language (UML). It helps you adapt to changing customer requirements, improves productivity and reduces time-to-market with advanced validation and simulation features.

Rational Rhapsody Designer for Systems Engineers provides:
Simulation and model execution—to validate system behavior early
Requirements analysis and traceability—to perform trade studies and design structural and behavioral aspects of your systems.
Team collaboration—to manage the complexity of developing consistent designs across different environments.
Visual development—to capture designs graphically, using industry-standard SysML and UML, or with Domain Specific Languages (DSL) such as AUTOSAR.
Lifecycle support and add-on software—to enable full product lifecycle development through integration with IBM Rational products and other software.

TBreq

TBreq provides the features to assist with quality and certification standards for multi-level and bi-directional requirements traceability and impact analysis.

TBreq is the easiest way to link requirements, design, development, testing and verification workflow with your Requirements Management tools and your design and development tool chain.

TBreq, through its integration with the LDRA tool suite which includes LDRA Testbed® and TBrun® (unit testing component), is a unique solution that can help your team overcome the challenges of allocating requirements to team members, mapping requirements to design and source code, linking test cases to requirements and the generation of verification reports.

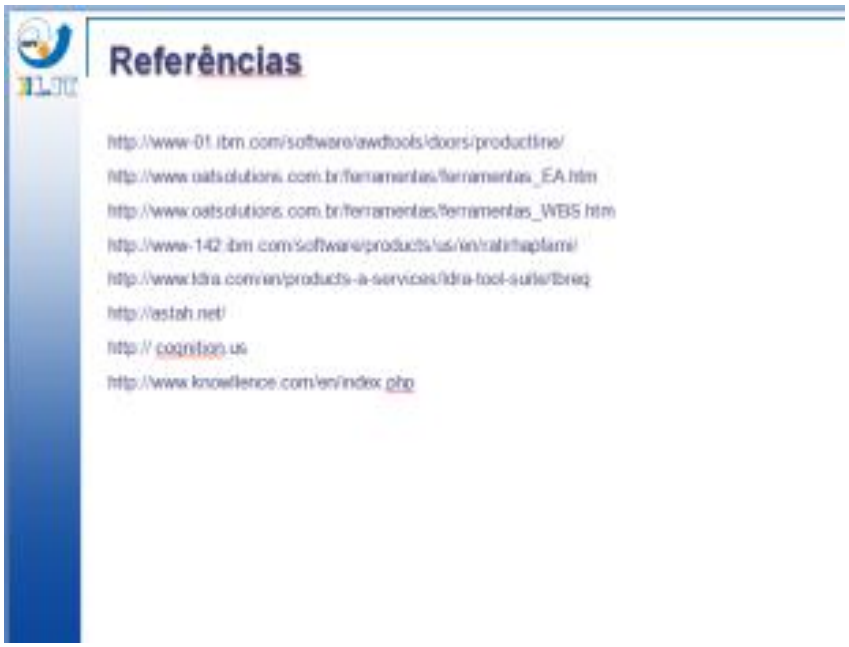
TBreq

TBreq interfaces directly with your management tool (IBM® Rational® DOORS®, IBM® Rational® RequisitePro®, Visure IRQA, Microsoft® Word or Microsoft® Excel) to ensure traceability across your software lifecycle and the completeness of your requirements coverage.

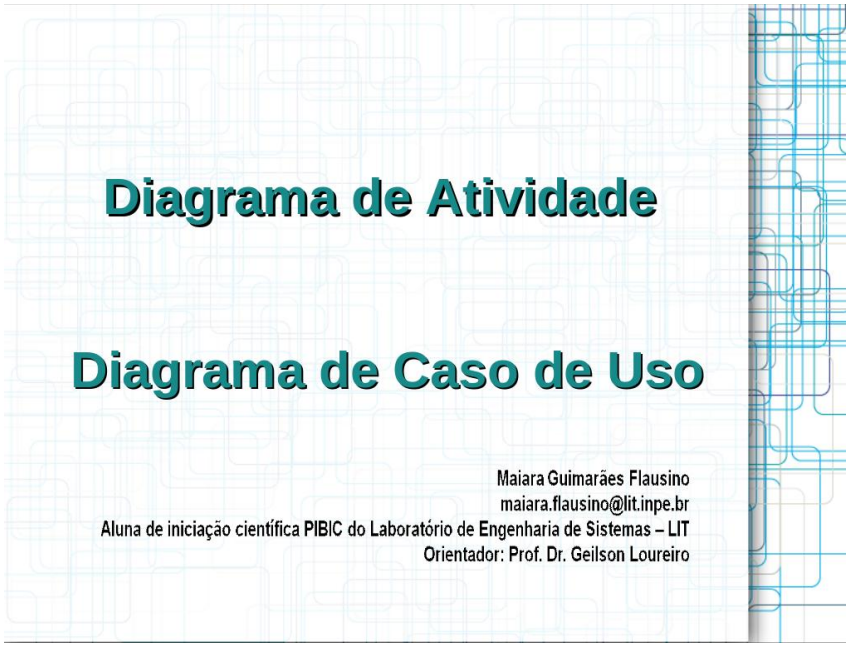
Within the LDRA tool suite, TBreq creates test specifications and executable test cases directly from requirements. Test results are automatically returned to the Requirements Management tool to provide "round-trip requirements traceability verification."

A key feature of TBreq is its ability to capture requirements (high-level, derived and low-level) from any management tool and source; while providing an intuitive interface for traceability, test case generation and requirements verification. All of these features combine to make **TBreq** the most effective solution for Requirements Management on the market today.


WBS Chart Pro
O WBS Chart Pro pode ser usado para gerar gráficos WBS diretamente a partir de planos já existentes do Microsoft Project. O WBS Chart Pro usa o diagrama criado no Microsoft Project para gera uma visão hierárquica dos dados. Uma barra de ferramentas WBS pode ser instalada no Microsoft Project para transferência simples de dados. Conforme você organiza e muda seu projeto do Microsoft Project, um gráfico WBS dos dados está a apenas um botão de distância.
A interface entre o Microsoft Project e o WBS Chart Pro é completamente transparente e bidirecional. Quando você cria um gráfico WBS de um plano já existente do Microsoft Project, todas as mudanças feitas no gráfico WBS são imediatamente refletidas no plano do Microsoft Project. Isto significa que você pode acrescentar, apagar, reorganizar e atualizar tarefas no seu plano do Microsfot Project usando o gráfico WBS. Nenhum outro software de gráfico no mercado hoje permite este nível de funcionalidade para gerencia seus planos do Microsoft Project.
Além disso, os gráfico WBS podem ser criado primeiro o WBS Chart Pro e depois transferidos diretamente para o Microsoft Project. Quando um gráfico WBS é tranferido para o Microsoft Project, todas as informações de tarefas e recursos colocados no gráfico WBS são automaticamente transferidas para o Microsoft Project onde mais agendamentos podem ser realizados. Você pode então continua a mudar entre o Microsoft Project e o WBS Chart Pro conforme achar necessário.


Referências
http://www-01.ibm.com/software/awdtools/doors/productline/
http://www.oatsolutions.com.br/ferramentas/ferramentas_EA.htm
http://www.oatsolutions.com.br/ferramentas/ferramentas_WBS.htm
http://www-142.ibm.com/software/products/us/en/ratirhapfami/
http://www.tdra.com/en/products-a-services/tdra-tool-suite/tbreq
http://astah.net/
http://cognition.us
http://www.knowllence.com/en/index.php

ANEXO K - Apresentação de trabalho de aula ITA 2011: Diagrama de atividade e Diagrama de caso de uso



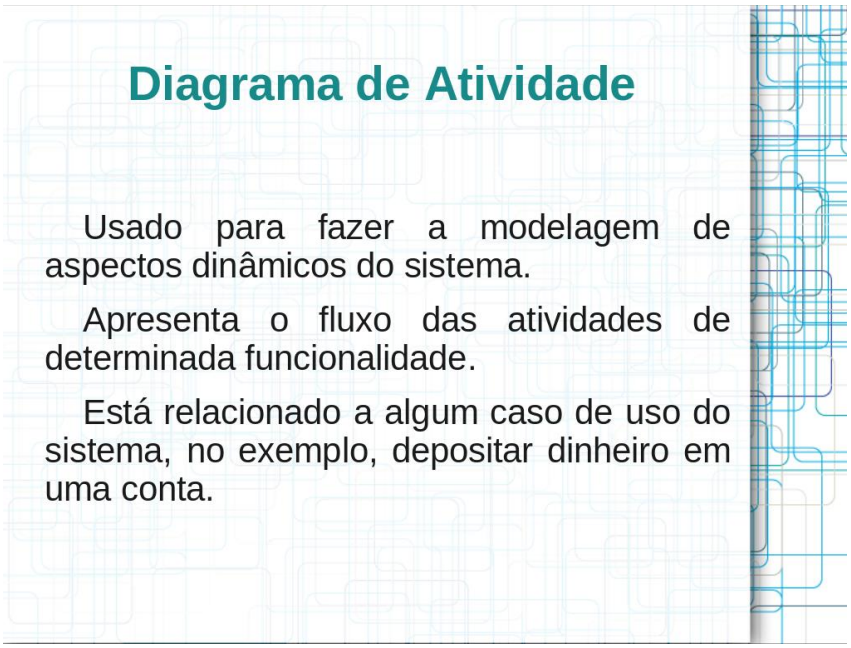


Diagrama de Atividade

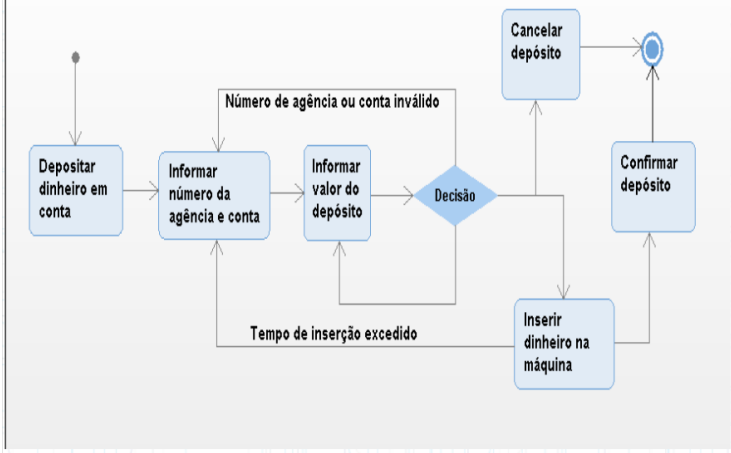


Diagrama de caso de uso

Assim como o de atividade, este diagrama também modela aspectos dinâmicos do sistema, no exemplo mostra as funcionalidades do sistema e a interação delas com os atores.

"Documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo."
Ivar Jacobson

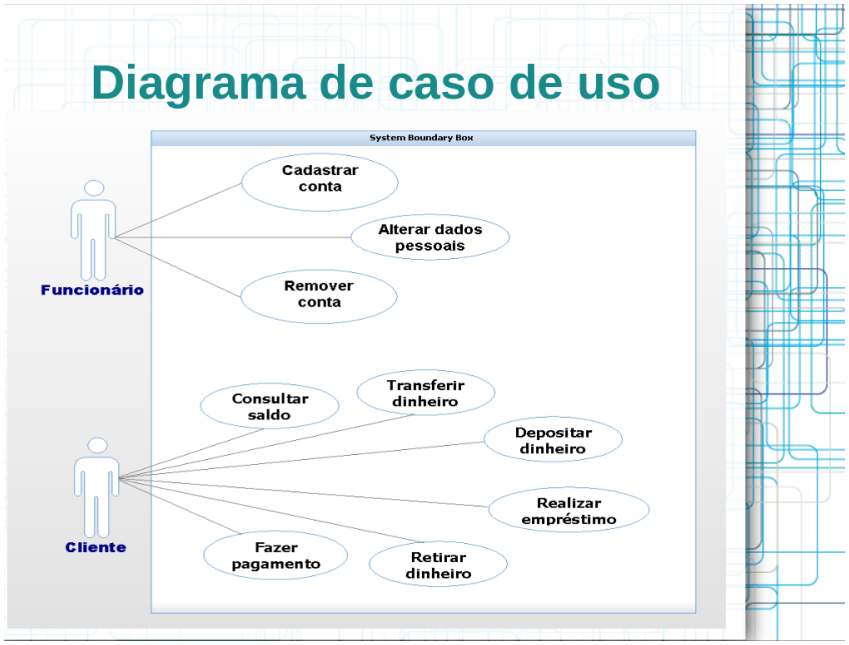
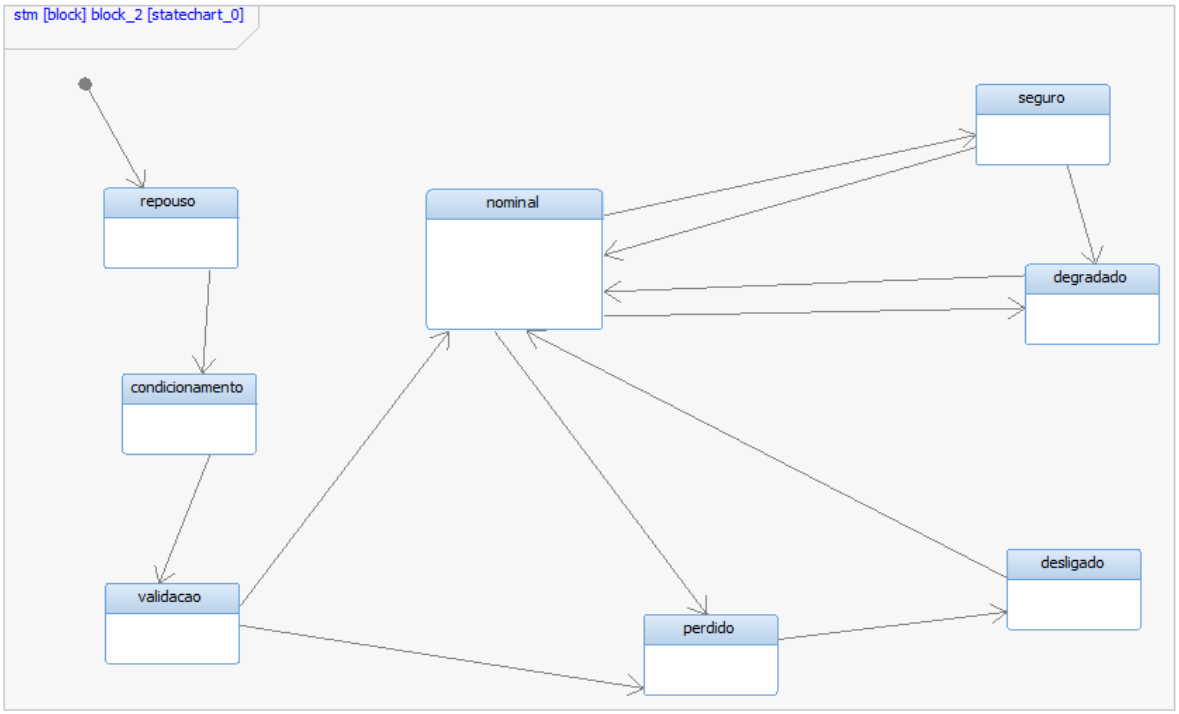
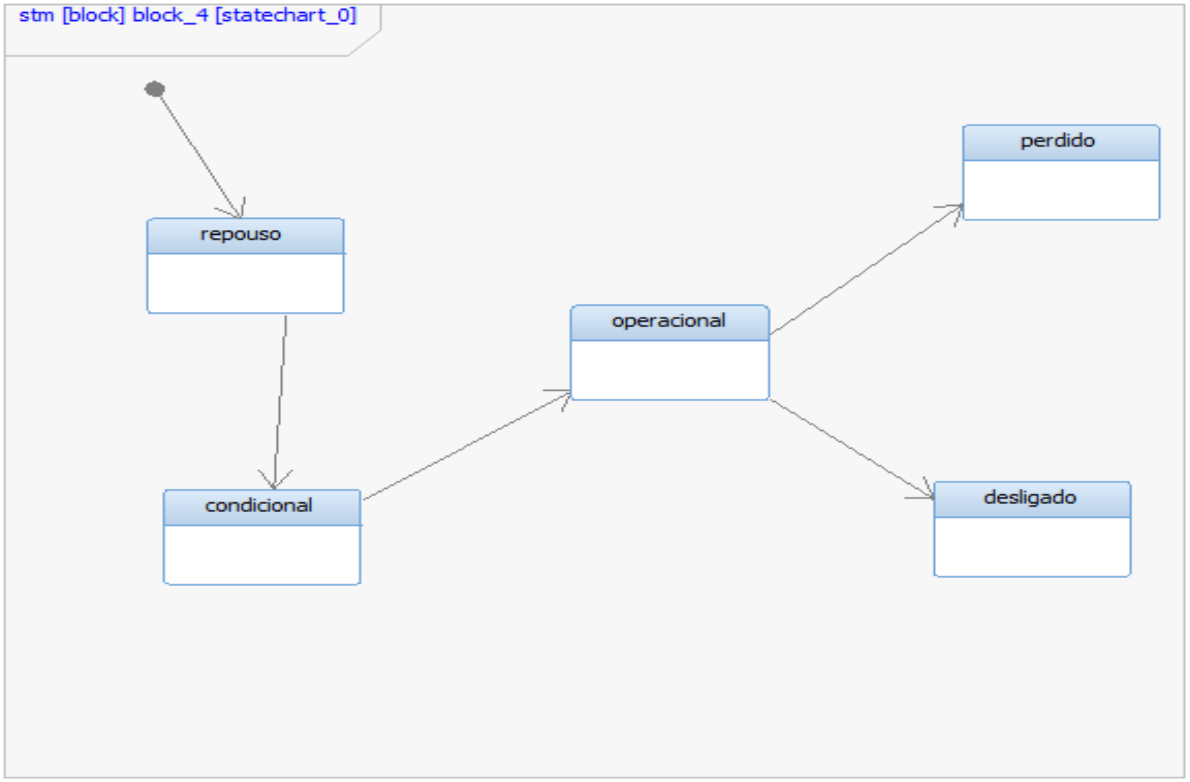
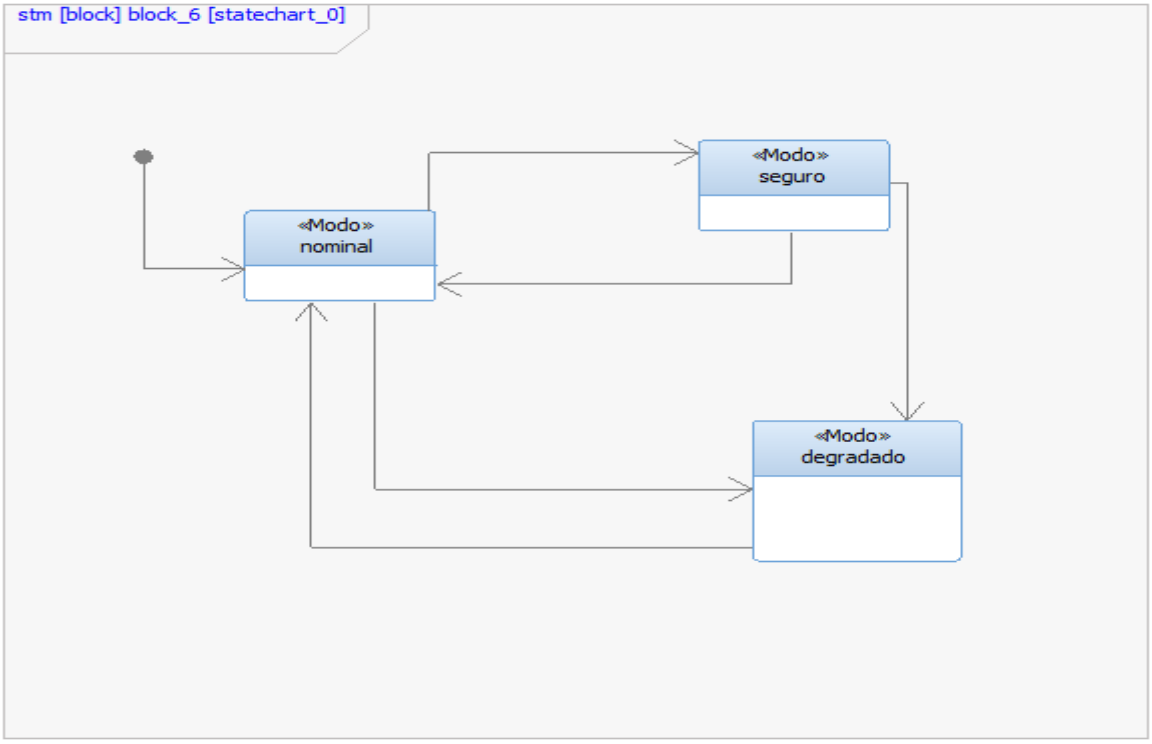


Diagrama de caso de uso
System Boundary Box
Cadastrar conta
Alterar dados pessoais
Remover conta
Funcionário
Consultar saldo
Transferir dinheiro
Depositar dinheiro
Realizar empréstimo
Fazer pagamento
Retirar dinheiro
Cliente

ANEXO L – Diagramas de máquina de estado da linguagem SysML do AESP-14







Estado	Condição	Próximo Estado	Ação
repouso	ejeção do satélite	condicionamento	ligar satélite
			enviar morse
condicionamento	30' após separação	operacional	habilitar potência máxima de transição
	perda de rastreabilidade	perdido	___
operacional	perda de rastreabilidade	perdido	___
	TC desligar	desligado	desligar satélite
perdido	esgostamento da bateria	desligado	___
desligado reentrada			
nominal	nível de bateria baixo	seguro	desativar payload/desabilitar TM dados científicos
	falha dedectada	degradado	desligar o subsistema em falha
degradado	TC reset	nominal	ligar o subsistema desligados
seguro(econômico)	nível de bateria normal	nominal	habilitar a ativacão do payload;habilitar TM dados científicos
	falha dedectada	degradado	desligar o subsistema em falha