INTEGRANDO SYSML E MODEL CHECKING PARA V&V DE SOFTWARE CRÍTICO ESPACIAL

Eduardo Correia da Silva, ecorreia@ita.br
Emilia Villani, evillani@ita.br
Instituto Tecnológico de Aeronáutica – ITA.
Praça Marechal Eduardo Gomes, 50 - Vila das Acácias
CEP 12.228-900 – São José dos Campos – SP – Brasil

Resumo. Este artigo apresenta uma abordagem sistemática para especificação, V&V de requisitos de *software* embarcado espacial em concordância com as normas européias para o setor espacial definidas pela ECSS. Esta proposta é baseada na linguagem de modelagem SysML, na utilização de autômatos para modelagem do sistema e na abordagem de *model checking* para V&V. A abordagem proposta é aplicada a um estudo de caso que consiste em uma plataforma aerosuspensa que simula o controle de atitude em um eixo de um satélite. Sob a plataforma estão instalados um sensor de atitude (giroscópio), um atuador (roda de reação) e um computador de bordo.

Keywords*: SysML, UML, Model Checking, Verification and Validation*

1. INTRODUÇÃO

Este artigo aborda a utilização de modelos para validação e verificação (V&V) de requisitos de software para aplicações espaciais, inserido no contexto Programa ITASAT, financiado pela Agência Espacial Brasileira (AEB) e desenvolvido através de cooperação entre o Instituto Nacional de Pesquisas Espaciais (INPE) e o Instituto Tecnológico de Aeronáutica (ITA). O objetivo do Programa ITASAT é a capacitação de recursos humanos, a divulgação de conhecimento e a geração de inovação tecnológica para área aeroespacial através de conceituação, projeto e desenvolvimento de satélites de pequeno porte e de pesquisas aplicadas relacionadas às necessidades nacionais. Um dos pontos de inovação deste Programa é o desenvolvimento de um computador de bordo (OBC – On Board Computer) que integra as funções de gerenciamento do satélite (conhecidas como housekeeping) e controle de atitude e órbita. Esta configuração é denominada ACDH (Attitude Control and Data Handling) e diferencia-se da configuração utilizada normalmente nos satélites desenvolvidos pelo INPE, onde existe um computador específico para housekeeping (OBDH – On Board Data Handling) e outro para controle de atitude e órbita (AOCS – Attitude and Orbit Control System).

Como resultado da configuração unificada do ACDH, torna-se necessário a utilização de processadores embarcados de alto desempenho, como o ERC32. Quando comparado com processadores mais simples, as limitações de tamanho de código e tempo de execução, que geralmente impunham a codificação de rotinas diretamente em Assembler, deixam de direcionar o desenvolvimento do software embarcado. Tem-se então um novo paradigma de desenvolvimento onde se torna viável a utilização de um sistema operacional de tempo real e a incorporação de novas funcionalidades no software aplicativo, que, conseqüentemente, terá maior complexidade.

Este novo paradigma requer o uso de métodos sistematizados para concepção e projeto do software embarcado, o que inclui, entre outras coisas, o uso de técnicas de modelagem e abordagens de validação baseada em modelos. Este é o tema deste artigo, que aborda a relação da utilização da linguagem de modelagem SysML (Systems Modeling Language) ao longo do ciclo de desenvolvimento do sistema e do software, e em concordância com as exigências das normas ECSS (European Cooperation on Space Standardization).

O objetivo deste artigo é apresentar uma abordagem sistemática para especificação, V&V de requisitos de *software* embarcado em concordância com as normas ECSS. Esta proposta é baseada na linguagem de modelagem SysML, na utilização de autômatos para modelagem do sistema e na abordagem de *model checking* para V&V.

Este artigo está organizado da seguinte forma. A Seção 2 introduz a SysML como linguagem de modelagem. A Seção 3 apresenta o método proposto, que caracteriza a contribuição deste artigo. A Seção 4 apresenta um estudo de caso, enquanto que a Seção 5 discute algumas conclusões e detalha os trabalhos futuros.

2. SysML e a Modelagem de Sistemas

2.1. Os Diagramas da SysML

SysML é uma linguagem de modelagem gráfica de uso geral para especificar, analisar, projetar, e verificar sistemas complexos que podem incluir hardware, software, informação, pessoal, procedimentos, e facilidades . Especificamente, essa linguagem fornece representações gráficas com significado semântico para modelar requisitos, comportamento, estrutura, e integração do sistema com uma análise da engenharia em larga escala [Robinson, Scopel, Rheinheimer, Martins, Pinto, Giraffa, Menezes, 2004]. A SysML representa um subconjunto de UML 2.0, com as extensões necessárias para satisfazer as necessidades da Engenharia de Sistemas.

Historicamente, a SysML nasceu a partir de uma iniciativa do *International Council on Systems Engineering* (INCOSE) para customizar a UML para aplicações de Engenharia de Sistemas. O INCOSE e a OMG (*Object Management Group*, responsável pela UML), criaram em 2001 um grupo de trabalho que definiu os requisitos de uma linguagem de modelagem para Engenharia de Sistemas [OMG, 2007]. Estes requisitos foram publicados em 2003 como uma chamada para propostas (*UML for Systems Engineering Request for Proposal* - UML for SE RFP). Foi então organizado o SysML Partners, um grupo de trabalho composto por representantes do setor industrial e produtores de ferramentas CASE. Este grupo iniciou a elaboração da SysML *Open Source*, uma linguagem que respondesse aos requisitos especificados na SE RFP. A versão *draft* da SysML foi publicada em 2004, enquanto que a primeira versão, SysML 1.0, foi proposta no final de 2005.

A organização da SysML em diagramas está representada na Figura 1.

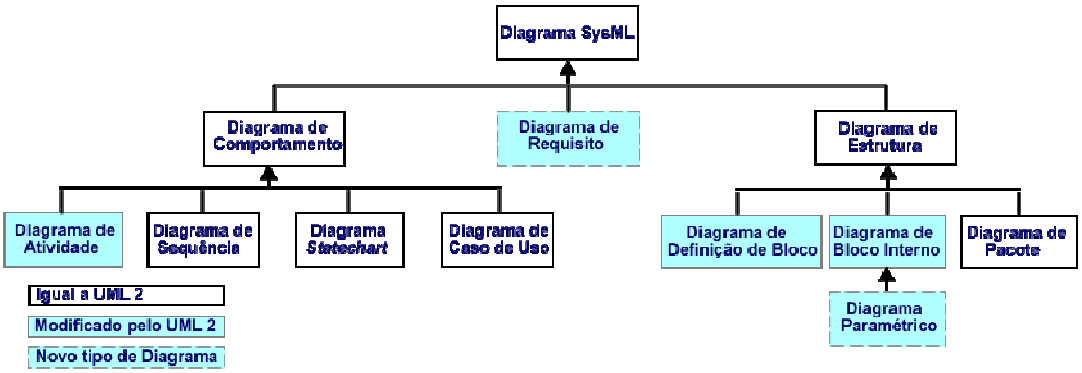

Figura 1. Diagramas da SysML, adaptado de (Herzog, Pandikow, 2005).

O "bloco" é a unidade básica da estrutura em SysML e pode ser usado para representar hardware, software, instalações, recursos, pessoal, ou qualquer outro elemento do sistema. A estrutura é representada por três tipos de diagramas: diagrama de definição de blocos (block definition diagram), que descreve a estrutura do sistema no que se referem aos componentes, suas propriedades, operações e relações e é equivalente ao diagrama de classes da UML; digrama interno de bloco (internal block diagram), que descreve a estrutura interna de um componente, incluindo suas partes e conectores; e diagrama de pacotes (package diagram), herdado da UML e apresenta a organização em pacotes dos modelos que compõem um sistema.

Um quarto diagrama também pode ser considerado como diagrama de estrutura, o diagrama paramétrico (parametric diagram), que representa restrições em valores de propriedade do sistema tais como desempenho, confiabilidade, e propriedades maciças, e serve como meio para integrar os modelos das fases de especificação e projeto com modelos da fase de análise.

O comportamento do sistema é representado por quatro tipos de diagramas herdados da UML: diagrama de atividades (activity diagram), que representa o fluxo dos dados e o controle entre atividades; diagrama de seqüência (sequence diagram), que representa a interação entre partes colaborativas de um sistema; diagrama de estado (statechart diagram), que descreve as transições de estado e respectivas ações que o sistema (ou parte do sistema) executa em resposta a eventos; e diagrama de casos de uso (use case diagram), que fornece uma descrição em alto nível das funcionalidades do sistema na forma de casos de uso.

Além da estrutura e comportamento, a SysML inclui uma construção gráfica para representar requisitos baseados em texto e para relacioná-las a outros elementos do modelo. O diagrama de requisitos (requirement diagram) ilustra os requisitos do sistema e suas relações com outros componentes. Ele fornece uma ponte entre as ferramentas de gerência típicas da Engenharia de Requisitos e os demais modelos do sistema.

2.2. SysML e a ferramenta Rhapsody

Ferramentas de suporte ao desenvolvimento nas diferentes fases do ciclo de vida do software, bem como na sua gerência e documentação, são conhecidas como ferramentas CASE (Computer-Aided Software Engineering). Estas ferramentas suportam desde a especificação e análise dos requisitos, até o projeto e implementação do sistema, com uso de bases de dados e interfaces gráficas e textuais [BARRERÉ, PRADO, BONAFE, 2005]. Ferramentas CASE automatizam diversas tarefas do desenvolvimento baseado em modelos (MDD) e permitem a verificação automática de consistência entre as diversas representações (diagramas) de um sistema.

Neste trabalho, a ferramenta CASE utilizada é o Rhapsody, software desenvolvido pela indústria de softwares I-Logix, recentemente comprada pelo grupo TELELOGIC. O Rhapsody utiliza como linguagem de modelagem a SysML e, atualmente, permite gerar automaticamente códigos para aplicações em C, C++, ADA e Java. É possível programar códigos diretamente nas linguagens anteriormente mencionadas e nos modelos SysML.

2.3. SysML e o Desenvolvimento de Sistemas Embarcados

A adoção da SysML no desenvolvimento de sistemas embarcados em diversas áreas de atuação é ilustrada pelos trabalhos científicos publicados neste tema. Estes trabalhos motivam e fundamentam a escolha da SysML como linguagem de modelagem para sistemas embarcados de aplicações aeroespaciais. Alguns exemplos são brevemente introduzidos a seguir.

Em [Rao, Dhadylla, Jones, McMurran, 2006] apresenta-se o uso da SysML no desenvolvimento de um sistema de informação para uma aplicação da industria automotiva. Este trabalho verifica os benefícios da utilização de técnicas de modelagem e refinamento dos requisitos do sistema ao longo de todo o ciclo de desenvolvimento. Este estudo de caso investiga a capacidade de controle dos requisitos de um sistema, discutindo os possíveis benefícios da SysML na redução dos esforços para criação, manipulação e verificação dos requisitos. Em particular analisa-se as vantagens SysML em confronto com uma abordagem funcional baseada nas ferramentas CASE MATLAB/Simulink/Sateflow, sendo esta última tradicionalmente usada para o desenvolvimento de software embarcado no ramo automotivo.

Bassi, Fantuzzi, Bonfé [2006] baseiam-se na SysML para definir um processo de projeto interativo e hierárquico, voltado para sistemas complexos. O detalhamento do sistema é acompanhado de atividades de validação ao longo de todas as fases do ciclo de desenvolvimento e é suportado pelos recursos de modelagem hierárquica e detalhamento sucessivo da SysML. A abordagem usada para validação é considerada como ponto de partida para o trabalho deste artigo.

O trabalho desenvolvido em Linhares, Silva, Oliveira [2006] também fundamenta a motivação de uso da SysML neste artigo. Este trabalho apresenta a modelagem de um sistema de automação industrial que inclui pessoas, hardware, software e outros elementos, possibilitando a integração de equipes de projeto de diversas áreas, tais como software, elétrica, mecânica e outros. Os modelos desenvolvidos ao longo do projeto compreendem o sistema supervisório de uma planta industrial, o sistema físico controlado, dispositivos de controle, a lei de controle utilizada, entre outros. Neste caso, a SysML atua de forma a proporcionar uma elaboração mais prática e eficiente dos diagramas de estrutura, análise e implementação do sistema supervisório.

Muitos dos trabalhos publicados relativos à SysML concentram-se na modelagem de requisitos, uma vez que esta é uma das principais disciplinas da Engenharia de Sistemas. Um exemplo é a proposta apresentada em [Colombo, Bianco, Lavazza, Coen-Porisini, 2008], onde discute-se a utilização em conjunto da SysML com Problem Frames (PF). PF é uma abordagem rigorosa utilizada para modelagem de requisitos que tem alto potencial de melhoria no processo de desenvolvimento de software, mas tem a desvantagem de não possuir uma notação gráfica intuitiva e de fácil compreensão. A proposta deste trabalho é, então, prover esta notação usando a SysML, possibilitando assim sua aplicação para problemas industriais.

Por meio dos exemplos citados verifica-se que a SysML proporciona uma abordagem clara, objetiva e eficiente de modelagem de sistemas, ou de soluções integradas de desenvolvimento de sistemas complexos, em âmbito industrial, científico e tempo real, fornecendo suporte para atividades de verificação e validação de sistemas.

Por outro lado, conforme observado por alguns dos trabalhos apresentados (Bassi; Fantuzzi; Bonfé, 2006) (Colombo, Bianco, Lavazza, Coen-Porisini, 2008), um dos pontos fracos da SysML é a ausência de ferramentas para verificação formal dos requisitos. Neste sentido, este artigo visa-se suprir esta deficiência por meio da incorporação, no processo de desenvolvimento, da modelagem baseada em autômatos e de *model checking*. A descrição destas ferramentas é apresentada no próximo capítulo.

3. MÉTODO PROPOSTO

3.1 Normas da ECSS

Este trabalho apresenta diversos conceitos relacionados ao desenvolvimento de software embarcado para produtos espaciais, utilizando como principal referência as normas da ECSS, para tanto é importante perceber alguns pontos bases da ECSS.

A European Cooperation for Space Standardization (ECSS) é uma organização criada com o objetivo de desenvolver e manter um conjunto único de normas para o uso em qualquer projeto europeu da área espacial.

De acordo com a ECSS-P-00A Standardization Policy, as normas devem promover a melhoria contínua dos métodos e técnicas usados em projetos espaciais, prevenindo trabalho desnecessário, atribuindo experiência de projetos anteriores e outras fontes adequadas sistematicamente ao sistema,e estas são divididas e caracterizadas em domínios, os quais para este trabalho se refletem nos domínios da Engenharia e da Qualidade, em particular, consideram-se as normas ECSS-E-10 System Engineering, ECSS-E-40 Software Engineering e ECSS-Q-80 Software Product Assurance.

Um ponto importante da utilização das normas da ECSS é a definição dos requisitos do sistema, onde o processo da Engenharia de Requisitos pode ser organizado de diversas formas. [Cysneiros, Leite, 1998] identificam três fases: elicitação, modelagem e análise. Já Soares [2007] divide o processo em quatro atividades principais: identificação, análise e negociação, especificação e documentação, e validação. Neste trabalho, de acordo com instruções da ECSS-E-

10 Part 1B Requirements and Process, o processo deve compreender as seguintes atividades básicas: captura, análise e validação, alocação, e manutenção.

A Engenharia de Requisitos aplicada neste trabalho é englobado pela método proposto para o desenvolvimento de software crítico embarcado e de tempo real. O método apresenta-se definida em fases, caracterizando-se por uma abordagem proposta por 10 passos, que podem ser observados no ciclo de vida de software e da missão da ECSS, conforme ilustrado na Figura 2.

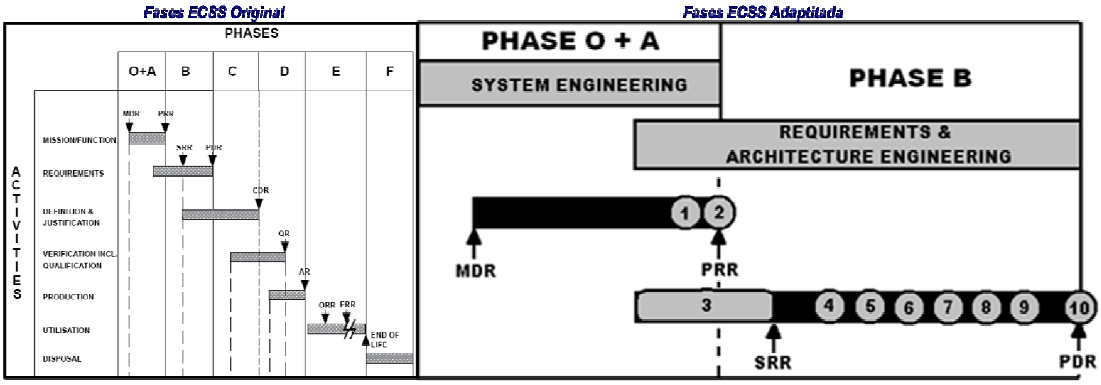

Figura 2. Comparativo entre Fases das Normas ECSS Original e Adaptada.

Os passos 1 e 2 estão localizados ao final da Fase A, finalizado com a PRR (Preliminary Requirement Review). O passo 3 engloba o início da Fase B e é finalizado com a SRR (System Requirement Review). Os demais passos se enquadram ao longo da Fase B, que termina com a PDR (Preliminary Design Review).

A abordagem proposta é composta dos seguintes passos:

- PASSO 1: Definição das interfaces do sistema em alto nível.
- PASSO 2: Definição das principais funcionalidades do sistema.
- PASSO 3: Elaboração da lista de requisitos do sistema.
- PASSO 4: Detalhamento da lista de requisitos do software embarcado.
- PASSO 5: Elaboração do diagrama de requisitos.
- PASSO 6: Proposta de solução em statechart.
- PASSO 7: Conversão do modelo para autômatos.
- PASSO 8: Verificação por simulação.
- PASSO 9: Verificação por model checking.
- PASSO 10: Revisão da especificação de requisitos e dos modelos em statechart.

3.2 Introdução aos Autômatos Temporizados e UPPAL

Apesar de não ter sido concebida com o propósito formal de modelagem, podemos estabelecer um dos pontos fracos da SysML como uma ausência de um formalismo matemático associado aos diagramas que permita a verificação formal de propriedades nos modelos desenvolvidos. Para suprir esta necessidade este trabalho propõe o uso de autômatos, que são largamente utilizados para modelagem, simulação e verificação de sistemas em diferentes áreas.

Os autômatos permitem a modelagem de sistemas dirigidos a eventos discretos, isto é, sistemas cujo comportamento dinâmico é regido pela ocorrência de eventos considerados instantâneos, que modificam o estado do sistema. Entre a ocorrência de eventos o estado do sistema não é alterado.

A motivação para escolha de autômatos é sua similaridade com os modelos elaborados em statechart, facilitando a conversão entre ambos. Além disso, a possibilidade de incorporação de tempo nos modelos em autômatos permite a verificação de requisitos de tempo real, um ponto essencial no desenvolvimento de software embarcado para aplicações espaciais. No caso do ACDH, uma simples variação no tempo de resposta a uma entrada pode gerar um encadeamento de problemas e resultar na perda total do satélite. A utilização de métodos formais para verificação da especificação de requisitos permite considerar todo o espaço de estados do modelo elaborado, garantindo assim a ausência de erros, inconsistências e especificação não completa.

Entre as ferramentas para suporte a modelagem, simulação e verificação de autômatos, este trabalho utiliza a ferramenta UPPAAL, a qual tem como principais vantagens a disponibilização de diversos recursos adicionais para modelagem (como a instanciação de templates), a possibilidade de incorporação de tempo no modelo e a utilização de model checking para verificação formal, permite que sejam submetidas para verificação propriedades definidas utilizando fórmulas CTL (Computational Tree Logic), utilizando a técnica de model checking, fornecendo resposta à quanto a veracidade da propriedade, ou a ocorrência de erro no processamento da fórmula. Desta forma, o UPPAAL complementa a utilização da ferramenta CASE no que ser refere a verificação formal de requisitos.

4. ESTUDO DE CASO

4.1. Simulador de ACDH

O estudo de caso considerado para aplicação da abordagem proposta é uma plataforma aerosuspensa que simula o controle de atitude em um eixo de um satélite. Sob a plataforma estão instalados um sensor de atitude (giroscópio), um atuador (roda de reação) e um computador de bordo. O computador tem a função de um ACDH (sigla em inglês de Controle de Atitude e Supervisão de Bordo). Além de estabelecer a comunicação com a estação de solo, trocando telemetrias e telecomandos, o computador de bordo deve ainda executar a malha de controle para manter a atitude definida pela estação de solo.

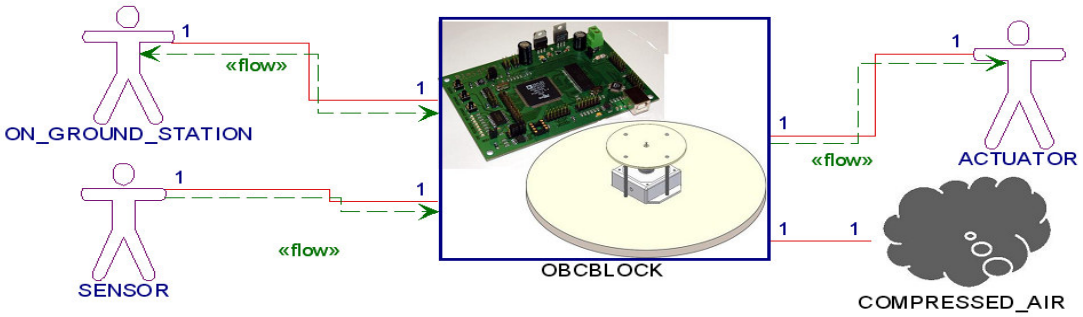

Figura 3. Diagrama de Blocos da Plataforma Aero suspensa.

2.2. Sistema e requisitos associados as normas ECSS

Para que um sistema espacial crítico se torne confiável, é importante que este atenda aos requisitos de desenvolvimento de software das normas da ECSS.

O processo de desenvolvimento inicia-se na a Fase O + A, através da definição da missão (MDR), onde todo o sistema tem como principal objetivo ou missão atuar de forma que a plataforma aerosuspensa venha a ser controlada pelo OBC (sigla em inglês para Computador de Bordo) e supervisionada pela estação de solo. O sistema deve receber informações relativas a manobras e executá-las. Os componentes envolvidos no sistema e suas funções ainda como parte da Fase O + A são:

- Computador de Bordo (OBC) – Centraliza e coordena toda e qualquer comunicação entre as outras entidades;
- Giroscópio (SENSOR) – Provê dados de posição que são adquiridos pelo OBC, necessários para atualização de dados do controle de atitude;
- Roda de Reação (ACTUATOR) – Atua de forma a deslocar inercialmente o sistema, permitindo a correção de posição e realização de manobras;
- Estação de Solo (ON_GROUND_STATION) – Supervisiona a plataforma de controle, ou seja, verifica o comportamento, atribui funções, encaminha e recebe dados que comprovem que o mecanismo da plataforma está obedecendo a todas as leis que lhes fora atribuída, podendo interferir a qualquer momento na mesma.

O sistema proposto é ilustrado no diagrama da figura 3. A plataforma aerosuspensa permite que o sistema desenvolva um grau de liberdade em seu eixo de rotação, assemelhando-se a um controle de atitude em um eixo. Neste os dados adquiridos através do elemento SENSOR alimentam uma lei de controle que permite que sinais de controle para o ACTUATOR sejam processados, deslocamento o sistema para a atitude desejada. Esta definição atende o Passo 1 da abordagem proposta.

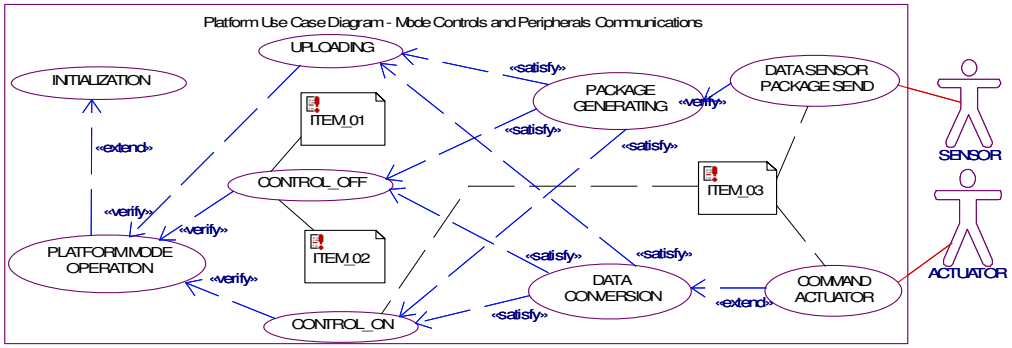

Figura 4. Diagrama de Caso de Uso para os modos de controle e conexões entre OBC e periféricos.

Como redução de escopo para enquadramento neste artigo, apenas algumas funcionalidades e requisitos são apresentados para ilustrar os próximos passos da abordagem proposta.

O Passo 2 consiste na elaboração do diagrama de caso de uso, que é apresentado na figura 4. Quando o OBC é inicializado, uma rotina é acionada adquirindo dados iniciais do sistema como a posição relativa inicial, lei de controle e valor de atitude. Três são os modos de operações do OBC: CONTROL_ON, CONTROL_OFF e UPLOADING. Quando em modo OFF, o sistema toma uma atitude de "stand by", ou seja, elementos SENSOR e ACTUATOR não são acionados ou controlados, e lei de controle estabelecida não é seguida. Quando em modo ON, os elementos SENSOR e ACTUATOR são acionados e controlados de acordo com a lei de controle. O modo UPLOADING atuada de modo semelhante ao modo OFF em relação aos elementos conectados porém permite substituir o trecho de software relativo a rotina de controle, que é enviado pela estação terra.

2.3. Definição dos requisitos e diagramas

No que se refere ao Passo 3, um ponto importante levantado pela norma ECSS-E-10 Part1B é a avaliação da criticidade de cada requisito, que é justificada pelo fato de que cada um difere em importância para o desenvolvimento e a operação do sistema nos termos de impacto no custo e no risco. Assim, para suportar o desenvolvimento do sistema minimizando risco e custo, deve-se avaliar a sensibilidade do sistema aos requisitos considerados críticos.

A fase 3 é seguida da na fase 4, onde os requisitos de sistema dão origem aos requisitos de software. Alguns exemplos de requisitos do OBC são apresentados na Tabela 1.

Tabela 1. Requisitos do Sistemas

ITEM	DESCRIÇÃO DO REQUISITO
01	When the platform is not following the attitude reference, the SENSOR shall be off.
02	When the platform is not following the attitude reference, the ACTUATOR shall be off.
03	When the platform is set to CONTROL_ON_MODE, the OBC shall turn the SENSOR and the ACTUATOR on.

Um importante diagrama proveniente da SysML e que permite verificar a interação entre os requisitos é o diagrama de requisitos, que engloba a fase 5. De acordo com esse tipo de diagramação é possível averiguar as formas e valores entre cada elemento, atribuindo principalmente a sua rastreabilidade.

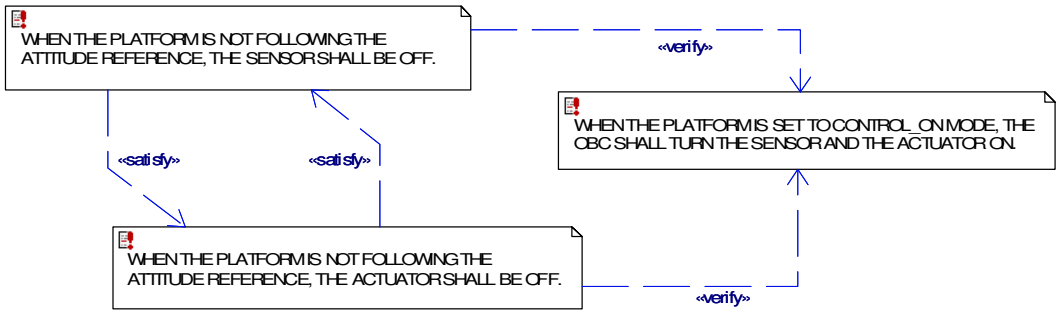

Figura 5. Diagrama de requisitos.

É possível notar que entre os requisitos apresentados na figura 5 existem interações que os classificam, permitindo que um processo de rastreabilidade destes seja de fácil entendimento. Observa-se que o requisito 3 proporciona o funcionamento dos demais através de verificação entre eles, ou seja, caso seja verificado que os requisitos 1 e 2 são satisfeitos, automaticamente o requisito 3 está satisfeito via verificação. A relação entre os requisitos 1 e 2 é de satisfação, ou seja, um requisito satisfaz o outro quando este é aceitável.

Vejamos pelo lado de funcionamento do sistema ACDH, onde o OBC quando não requisitado para modo de controle ligado, deve retornar para o sensor e o atuador a informação de inibição destes componentes, onde cada inibição satisfaz a outra e vice e versa. Quando verificado que ambos os elementos estão ativos, o requisito que denota a ativação do sensor e atuador quando o modo de controle ligado é satisfeito, ou seja, através da verificação de ativação de cada um dos outros dois requisitos, o terceiro é satisfeito.

Este diagrama é de extrema importância para a SysML, onde permite uma interação por completa do sistema, e não apenas do objeto controlador, ou seja, é possível explicitar estados e condições de usuários e interfaces que normalmente não seria permitida, proporcionando a aplicação dos mais diversos estados e atitudes do sistema.

Este diagrama e os demais da SysML são construídos utilizando ferramentas CASE (Computer-Aided Software Engineering). Estas ferramentas suportam desde a especificação e análise dos requisitos, até o projeto e implementação do sistema, com uso de bases de dados e interfaces gráficas e textuais [BARRERÉ, PRADO, BONAFE, 2005].

Ferramentas CASE automatizam diversas tarefas do desenvolvimento baseado em modelos (MDD) e permitem a verificação automática de consistência entre as diversas representações (diagramas) de um sistema.

Neste trabalho, a ferramenta CASE utilizada é o Rhapsody, software desenvolvido pela indústria de softwares I-Logix, recentemente comprada pelo grupo TELELOGIC.

O sistema ACDH apresentado neste artigo permite apenas a rotação em torno de um eixo. De acordo com o modo de controle do OBC do sistema, este deve estabelecer princípios que proporcionem seu funcionamento. Baseado nisto, apresenta-se na figura 6 o diagrama de estados referente ao modo de controle ligado do computador de bordo e que compõe o Passo 6 do processo de desenvolvimento de software adaptado à ECSS. Verifica-se que quando solicitado o computador disponibiliza informações para seus periféricos, ou seja, o sensor e atuador. Na plataforma utilizada – Rhapsody – os comandos para geração e envio de uma mensagem ou dado é o PORT (Elemento de saída)->GEN(Mensagem ou dado). O elemento de saída conectado ao sensor é denominado SENSOR_PORT, e o elemento de saída conectado ao atuador é denominado ACTUATOR_PORT.

O sensor reconhece basicamente as mensagens evSE_TON e evSE_TOFF para sua ativação e desativação respectivamente, e similarmente o atuador reconhece evACT_TON e evACT_TOFF. Notar que ao ser tratado como comando, foi designado que sempre a expressão "ev" será precedida da mensagem.

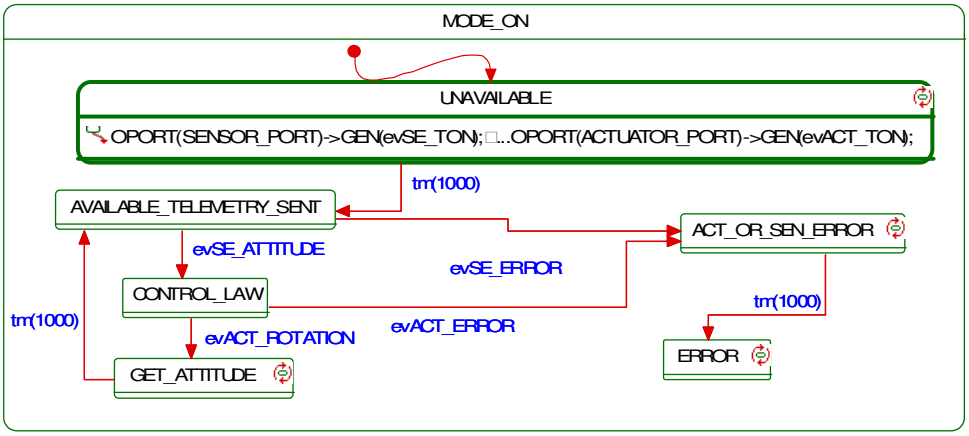

Figure 6. Diagrama de Estado em SysML

Com a ativação do estado MODE_ON, a unidade gera as duas mensagens evSE_TON e evACT_TON desprendidas através das portas de saídas do elemento, proporcionando desta forma a solicitação para ativação dos elementos correspondentes – SENSOR e ACTUATOR. Percebe-se que após determinado tempo programado, neste caso 1 segundo (tm (1000)), o estado atingido é o denominado AVAILABLE_TELEMETRY_SENT, que apresenta o inicio da comunicação entre o OBC e seus periféricos, bem como a transmissão para estação de solo via telemetria, caso que não iremos apresentar neste trabalho devido à necessidade de redução do escopo. Quando atingido este estado se prossegue com a comunicação entre os elementos, como podem ser observadas, para melhor entendimento, as mensagens evSE_ATTITUDE, evACT_ROTATION, evACT_ERROR e evSE_ERROR, que fazem parte das mensagens reconhecidas entre todos os elementos e geradas por estes. Com o exposto nota-se que com a ativação do modo de controle aplicado como demonstrado na figura 6, o requisito 3 é atingido.

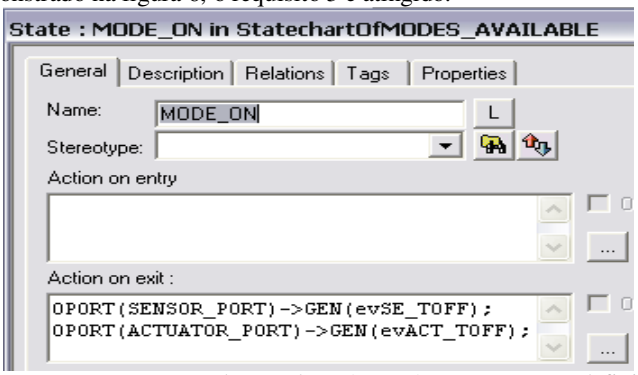

Figure 7. Representação do Estado MODE_ON e suas ações definidas

As mensagens de desativação para os elementos periféricos do OBC são classificadas como evSE_TOFF e evACT_TOFF, porém não são demonstradas no diagrama de estado do modo ativo de controle, porém a SysML permite que a entrada ou saída de um estado maior, neste caso MODE_ON, seja passível de aplicação de controle de mensagens

ou dados, de forma para o modelo apresentado, apenas quando de saída do estado ativado sejam geradas as mensagens de desativação dos elementos periféricos.

Note que a figura 7 apresenta em Action on exit, na caixa de diálogo State: MODE_ON, a mensagens pertinentes para a desativação. Desta forma fica simples perceber que não há como interferir nas condições apresentadas no diagrama de requisitos, ou seja, quando em estado ativo, necessariamente os elementos SENSOR e ACTUATOR estão ativados, porém em sua saída para estado desativado, ambos os elementos periféricos serão desativados, proporcionando assim uma validação nos requisitos propostos.

Como esta técnica de verificação de requisitos baseada na SysML é feita de uma maneira muito subjetiva, se faz necessária a sua verificação via métodos mais formais, para tanto o mesmo modelo, porém se reportando principalmente ao diagrama de estado, foi modelado em autômatos, que garantem uma equiparação de acordo e representa a Fase 7 do desenvolvimento de software.

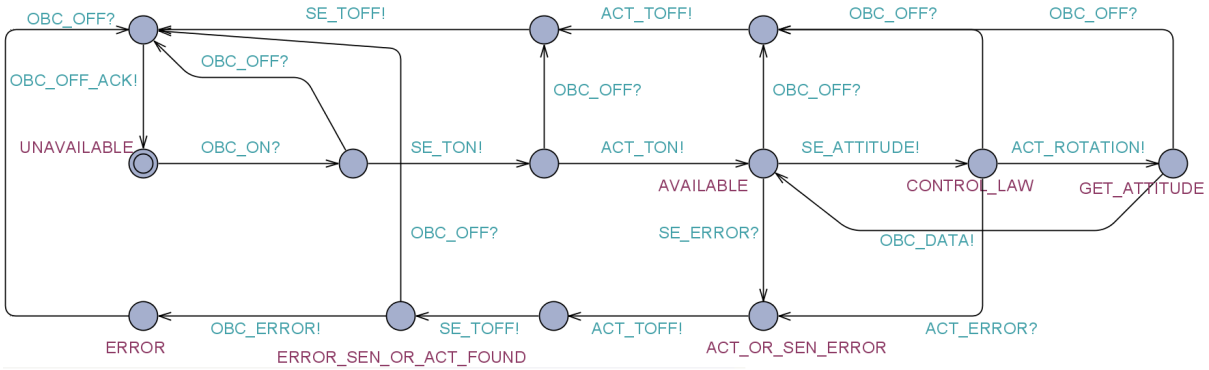

Figura 8. Diagrama Autômato para o modo de controle ativo.

Para ser possível uma equiparação mais correta e definida foram denominadas nomenclaturas idênticas entre os dois modelos, porém verifica-se um maior numero de estados quando em autômatos comparativamente ao exposto em SysML. Esta diferença se dá devido ao fato que na SysML é possível a geração de indefinidas mensagens ou comandos através de uma mesmo estado, o que não ocorre no autômato, que apenas permite uma mensagem de entrada para uma mensagem de saída. Desta forma, estados ou lugares que não possuem nomenclatura definida estão presentes apenas para que seja possível a geração da mensagem, porém fazem parte de um estado imaginário maior equivalente ao estado que pertencem as mensagens na visão do modelo da SysML.

Além da verificação formal ou *model* checking, correspondente ao Passo 9, é possível demonstrar a equivalência entre os modelos de acordo com os diagramas de seqüência disponíveis nos dois ambientes, caracterizando o Passo 8. Através da figura 9 estão definidos os estados atingidos durante instante de comunicação entre o OBC e seus periféricos no momento em que o modo de controle CONTROL_ON é ativado e desativado após alguns segundos. Percebe-se que o diagrama apresentado se equipara ao modelo apresentado na figura 8, representando através do tempo as alterações do comportamento do sistema.

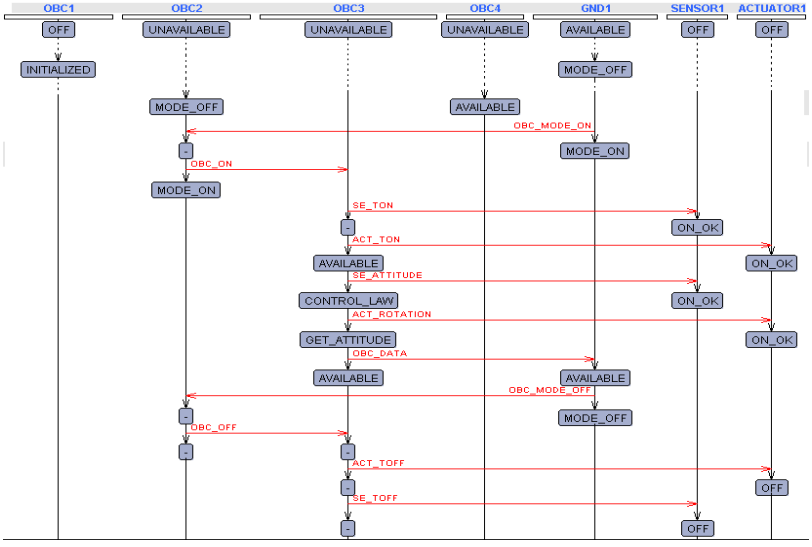

Figura 9. Diagrama de seqüência em autômatos.

Equivalentemente para o mesmo período, a figura 10 representa o diagrama de seqüência para o modelo em SysML, onde é possível perceber claramente que este possui as mesmas características do modelo em autômato, garantindo deste forma sua validação, uma vez que comprovadamente os requisitos e a disponibilidade no tempo do sistema são verificados através de técnica formal de *model checking* através do sistema de autômatos. Nota-se que todas as mensagens envolvidas neste período apresentado são equivalentes em tempo e estado, proporcionando uma verificação mais evidente entre os comportamentos de ambos os modelos, lembrando apenas que os elementos designados como OBC1, OBC2, OBC3 e OBC4 representam o único elemento demonstrado na figura 10 – OBC.

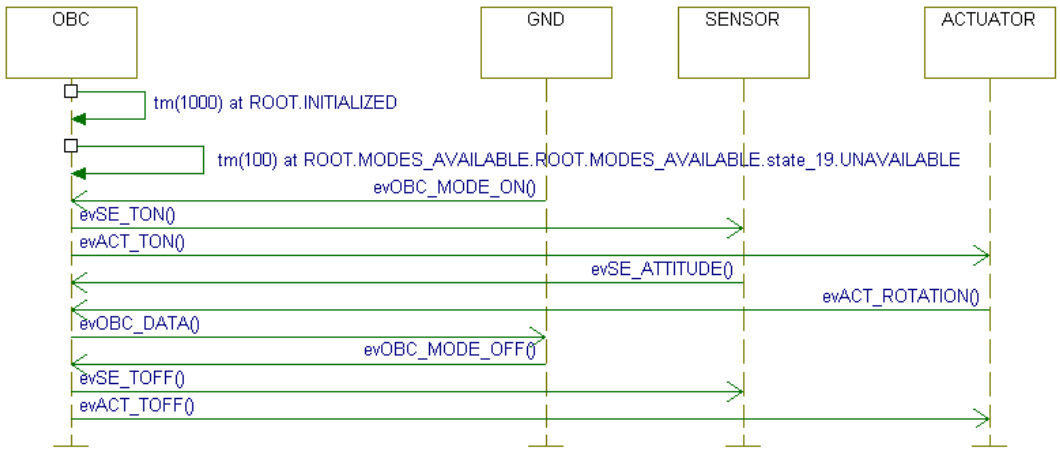

Figura 10. Diagrama de seqüência em SysML

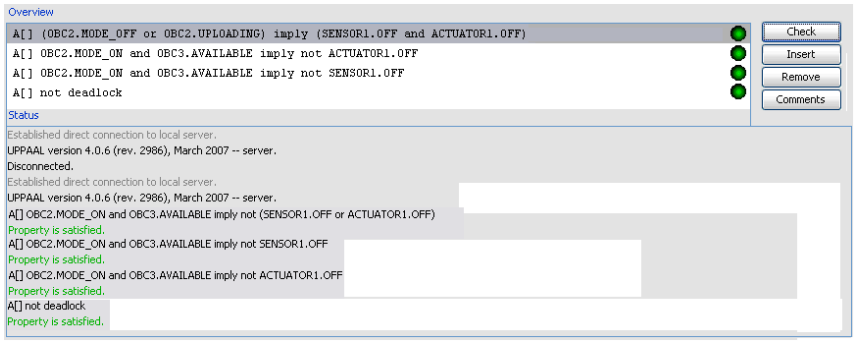

Figura 11. Verificação dos requisitos em plataforma UPPAL.

O UPPAAL proporciona um ambiente de verificação de requisitos através de “queries”. Para que seja melhor compreendido é importante notar que a nomenclatura do ambiente, onde a expressão “A[ ] p” é dita verdadeira se e somente se todo estado alcançável satisfaz p, e “imply” representa um conectivo entre os elementos a serem verificados. Como exemplo, a expressão “A[ ] X imply Y” é satisfeita quando o acontecimento de X implica o acontecimento de Y para todo estado alcançável. Desta forma é possível estabelecer uma conexão entre os elementos que compõem os requisitos propostos e desta forma pode-se verificar formalmente cada um deles, onde a satisfação de todos em um mesmo modelo define a validação do modelo.

Tendo em vista a declaração anterior temos apresentada na figura 11 à apresentação dos três requisitos para o modelo do ACDH traduzido para o ambiente de verificação.

Os modos de controle do OBC são três, MODE_ON, que determina o modo ativo da lei de controle, MODE_OFF, que determina o modo inativo da lei de controle e UPLOADING, que determina o modo stand by do sistema, criando e permitindo comunicação para atualização de dados para o OBC. Quando de saída do modo ativo, o OBC poderá estabelecer quaisquer outros modos, e para tanto a verificação dos requisitos apresentados nos itens 1 e 2, são alcançados de acordo com a saída do modo de controle ativo, como pode ser percebido através das seguinte condições:

- A[] (OBC2.MODE_OFF or OBC2.UPLOADING) imply (SENSOR1.OFF and ACTUATOR1.OFF), onde representa que quando não solicitado o modo de controle ativo, os periféricos são garantidos desligados, verificando desta forma o requisito 3;
- A[] (OBC2.MODE_ON and OBC3.AVAILABLE) imply not (SENSOR1.OFF), onde representa que quando em modo de controle ativo, o elemento SENSOR esteja em operação, podendo este estar nos estados ON_OK, isto é, esteja ativo e operando normalmente com o sistema, ou ON_FAULT, que significa que está ativo porém em falha, verificando o requisito 1;

- A[] OBC2.MODE_ON and OBC3.AVAILABLE imply not ACTUATOR1.OFF, onde representa que quando em modo de controle ativo, o elemento ACTUATOR esteja em operação, podendo estar nos estados ON_OK, isto é, esteja ativo e operando normalmente com o sistema, ou ON_FAULT, que significa que está ativo porém em falha, verificando o requisito 2.
- A[] not deadlock, onde esta importante verificação desprendida através do formalismo do autômato, garante que o sistema não entre em "deadlock", ou seja, não existe estado "N" onde o estado "N+1" não seja atingido. É importante notar que este requisito não é possível ser analisado de forma objetiva através da SysML, um estudo através da simulação e dos diagramas se faz necessária.

5. CONCLUSÃO

A proposta deste trabalho reúne duas técnicas de modelagem de sistemas temporizados, de modo que a utilização de verificação formal baseada em *model checking* e autômatos complementa a utilização da SysML no desenvolvimento de software para sistemas críticos embarcados e de tempo real. A abordagem proposta para integração das duas técnicas se baseia no ciclo de vida definido pelas normas européias ECSS.
As técnicas são aplicadas a estudo de caso desenvolvido no contexto do projeto ITASAT, buscando representar um sistema de ACDH de um satélite. Este estudo de caso consiste em uma plataforma aero suspensa e provida de uma lei de controle que rege o seu funcionamento, coordenando e controlando os periféricos conectados, dispondo desta forma de rotação em um grau de liberdade controlada pelo elemento central denominado de computador de bordo.

6. REFERÊNCIAS

Alur, R., Dill, D., A Theory of a Timed Automata, Stanford University, USA, 2006
Barrere, T., Prado, A., Bonafe, V., Case com Múltiplas Visões de Requisitos e Implementação Automática, SBES, Florianópolis, Brasil, 2005
Bassi, L. Secchi, C. Fantuzzi, C. Bonfe, M. An object-oriented approach to manufacturing systems modeling, IEEE-CASE, Shangai, China, 2006.
Beam, W.R., Systems Engineering: Architecture and Design. McGraw-Hill, NY, USA, 1990.
Bock, C. SysML and UML 2 Support for Activity Modeling, Published on line in Wiley InterScience, 2005.
Colombo, P. Del Bianco, V. Lavazza, L. Coen-Porisini, A. A Methodological Framework for SysML: a Problem Frames-based Approach, APSEC, Nagoya, Japão, 2007.
Cysneiros, L. M. ; Leite, J. C. S. P. Utilizando Requisitos Não Funcionais para a Análise de Modelos Orientado a Dados. I Workshop de Engenharia de Requisitos, Maringá, Brasil, 1998.
Friedenthal, S., Moore, A., Steiner, R., OMG Systems Modeling Language (OMG Tutorial), United States, 2007.
Hause, M. The SysML Modelling Language, Fifth European Systems Engineering Conference, Edinburgh, UK, 2006.
Herzog, E., Pandikow, A. SysML – an Assessment, Proceedings of the 15th INCOSE International Symposium,NY, USA, 2005.
Laplante, P.A. Real-Time Systems Design and Analysis: An Engineer's Handbook, IEEE Computer Society Press, Piscataway, USA, 1997.
Linhares, M.V., Silva, A.J. Oliveira, R.S., Empirical Evaluation of SysML through the Modeling of an Industrial Automation Unit, ETFA, Florianópolis, Brasil, 2006
Merz, S. Model Checking: A Tutorial Overview, London, UK, 2000.
Mokadem, H.B. Berard, B. Gourcuff, V. Roussel, J.-M. De Smet, O., Verification of timed multitask system with UPPAL, ETFA, Catania, Italy, 2005.
Pressman, R. S. Engenharia de Software. Makron Books, São Paulo, Brasil, 1995.
Rao, A.C. Dhadyalla, G. Jones, R.P. McMurran, R. Systems modelling of a driver information system - automotive industry case study, SMC, Los Angeles, USA, 2006.
Robinson, G., Scopel, M., Modelando Requisitos Especificados com Mapas conceituas através da UML-MC, Manaus, Brasil, 2004.
Soares, M.; Vrancken, J. Requirements specification and modeling through SysML. ISIC, Cambridge, USA, 2007
Tang, T., Yan, F. A formal Modeling and Verification Approach for Real-Time System, Columbus, USA, 2008.
The Object Management Group (OMG), referência site http://www.omg.org.
Wertz, J.R., Larson J.L., Space Mission Analysis and Design (Third Edition), Softback, London, UK, 1999.
Willard, Brian. UML for Systems Engineering. Amsterdam, The Netherlands, 2005.