UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RTSS: Uma Família de Técnicas de Leitura para Suporte à Inspeção de Modelos SYSML e SIMULINK

ERIK ACEIRO ANTONIO

ORIENTADORA: PROFA. DRA. SANDRA CAMARGO PINTO FERRAZ FABBRI

São Carlos - SP
Junho/2014

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RTSS: Uma Família de Técnicas de Leitura para Suporte à Inspeção de Modelos SYSML e SIMULINK

ERIK ACEIRO ANTONIO

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação, área de concentração: Engenharia de Software.
Orientadora: Dra. Sandra Camargo Pinto Ferraz Fabbri.

São Carlos - SP
Junho/2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária/UFSCar

A635rf

Antonio, Erik Aceiro.
RTSS : uma família de técnicas de leitura para suporte à inspeção de modelos SysML e Simulink / Erik Aceiro Antonio. -- São Carlos : UFSCar, 2015.
154 f.

Tese (Doutorado) -- Universidade Federal de São Carlos, 2014.

1. Validação, verificação e teste. 2. Sistemas embarcados. 3. SysML (Sistema de computador). 4. SIMULINK (Programa de computador). 5. Técnicas de leitura. I. Título.

CDD: 005.14 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"RTSS: Uma Família de Técnicas de Leitura para Suporte à Inspeção de Modelos: SYSML e SIMULINK"

Erik Aceiro Antonio

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Membros da Banca:

Profa. Dra. Sandra Camargo Pinto Ferraz Fabbri (Orientadora- DC/UFSCar)

Prof. Dr. José Carlos Maldonado (ICMC/USP)

Profa. Dra. Itana Maria de Souza Gimenes (UEM)

Prof. Dr. Glauco Augusto de Paula Caurin (EESC/USP)

Prof. Dr. Auri Marcelo Rizzo Vincenzi (UFG)

São Carlos
junho/2014

Dedico este trabalho

à minha querida esposa Milene,

à minha mãe Marina, meu irmão Douglas e

a meu pai Francisco (*in memorian).*

AGRADECIMENTOS

Em primeiro lugar, agradeço à Profa. Dra. Sandra Fabbri pelo empenho, dedicação e por todas as ocasiões em que preciosas sugestões foram dadas, especialmente, para o aprimoramento deste trabalho. Sou eternamente grato por todos seus valiosos ensinamentos que me ajudaram a enxergar mais adiante, e que de outro modo, este trabalho jamais poderia ter sido realizado.

Agradeço ao Prof. Dr. Glauco Caurin, por ter gentilmente disponibilizado inúmeras vezes, os seus alunos para aplicação dos estudos experimentais. Também agradeço as oportunidades de discussões que promoveram meu crescimento em uma área tão desafiadora como a de Sistemas Embarcados.

Também quero agradecer ao Prof. Dr. Fabiano Ferrari, pelo apoio dedicação e sugestões dadas durante as revisões dos artigos e trabalhos desenvolvidos.

Agradeço o Prof. Dr. José Carlos Maldonado pelo apoio, incentivo e valiosas sugestões dadas durante o desenvolvimento deste trabalho.

Agradeço à Profa. Dra. Itana Maria de Souza Gimenes e ao co-escritor da especificação SysML, Sr. Tim Weilkiens, por gentilmente terem cedido os diagramas utilizados nos exemplos utilizados neste trabalho.

A todos os colegas do LaPES, em especial, Anderson Belgamo, André Di Thommazo, Elis Hernandes, Guilherme Freire, Odair de Souza, Bárbara Castanheira e ao Rafael Rovina que me apoiaram e contribuíram, sobretudo com dicas valiosas.

Agradeço o Departamento de Computação (DC) da Universidade Federal de São Carlos, pela intra-estrutura e apoio durante o doutorado.

Agradeço o INCT-SEC e as agências de fomento FAPESP, CNPq e Capes pelo apoio financeiro e infra-estrutura cedida para realização deste trabalho.

Agradeço aos meus familiares, a meu irmão Douglas, minha mãe Marina e meu pai Francisco (*in memorian*), pelos primeiros incentivos.

Finalmente, um agradecimento muito que especial à minha esposa Milene, minha eterna companheira, que pelo seu inestimável carinho e apoio ao longo desses anos de convivência, sobretudo nos momentos mais difíceis, me fez acreditar que seria possível chegar ao final desta difícil, porém gratificante etapa. Obrigado, meu amor.

"*Imagination is more important than knowledge. For knowledge is limited to all we now know and understand, while imagination embraces the entire world, and all there ever will be to know and understand*."

Albert Einstein

RESUMO

Contexto: Em geral, os desenvolvedores de Sistemas Embarcados (SEs) iniciam tais sistemas a partir da elaboração dos diagramas mais próximos da fase de geração de código, como por exemplo, alguns diagramas SysML e o modelo Simulink. Apesar do amplo uso de tais diagramas pela comunidade de SEs, observa-se uma carência por atividades de Verificação e Validação (V&V). As normas de certificação existentes atuam, principalmente, no nível de código. **Objetivo:**definir uma família de técnicas de leitura – *Reading Techniques for SysML and Simulink* (RTSS) – que dê suporte à atividade de inspeção desses tipos dos diagramas, com o intuito de melhorar a qualidade do processo e do produto gerado, identificando defeitos tão logo os artefatos sejam construídos. **Metodologia**: as técnicas de leitura foram definidas por meio de um processo sistemático e dão suporte à inspeção de pares deartefatos. Elas levam em consideração algumas normas internacionais de certificação de SEs, além de elementos pertinentes às estruturas das linguagens SysML e Simulink. Além disso, para propiciar o uso das técnicas ao longo de um processo de desenvolvimento, utilizou-se como referência o processo SYSMOD, que adota diagramas SysML e modelos Simulink ao longo de suas fases. Para avaliar as técnicas RTSS foram conduzidos dois experimentos controlados e três exemplos de aplicação,à medida que as técnicas foram elaboradas.**Resultados:** os resultados mostraram que as técnicas são viáveis de serem utilizadas e que elas são capazes de identificar defeitos nos pares de artefatos para os quais elas foram projetadas. Além disso, constatou-se que defeitos que não foram identificados e corrigidos em uma determinada fase do desenvolvimento, foram propagados para fases subsequentes. **Conclusão:** Com base nesses resultados, pode-se concluir que as técnicas RTSS são capazes de detectar defeitos à medida que os artefatos são construídos, evitando que eles sejam propagados para fases futuras. Isso pode melhorar a qualidade do processo e do produto e pode também minimizar o retrabalho e o custo de se corrigir um defeito em fases adiantadas. Adicionalmente, as técnicas podem ser aplicadas mesmo que o processo SYSMOD não seja adotado, bastando que se tenha disponível o par de artefatos que é tratado em cada uma das técnicas.

Palavras-chave: Sistema Embarcado;SysML;Simulink;Técnica de Leitura;SYSMOD;Verificação e Validação; Processo de Desenvolvimento.

ABSTRACT

Context: Usually, developers of Embedded Systems (ESs) start the development from models next to the code generation phase, for example, SysML diagrams and Simulink models. Despite the whole use of these models by the ES community, there is a lack of Verification and Validation activities (V&V). The certification standards operate, mainly, on code level. **Aim:**to define a family of reading techniques – *Reading Techniques for SysML and Simulink* (RTSS) – that supports the inspection of these diagrams and models, aiming to improve the process and product quality through defects identification, as soon as artifacts are elaborated. **Method:** the reading techniques were defined based on a systematic process and they support pairs of artifacts. They take some international certification standards into account, as well as elements from the structure of SysML and Simulink languages. Besides, aiming to suggest the use of these techniques inside a development process, the SYSMOD process was took as reference, since it adopts SysML diagrams and Simulink models along its phases. For evaluating the RTSS techniques two controlled experiments and three case studies were conducted as the techniques were elaborated. **Results:** the results showed that it is feasible to use the techniques and that they are able to detect defects on the pair of artifacts for the ones they were designed. In addition, it was observed that defects that were not identified and corrected inside the phase they were generated, were propagated to the subsequent phases. **Conclusion:** Based on these results, we can conclude that the RTSS techniques are able to detect defects as the artifacts are elaborated, avoiding their propagation to further phases. This fact can improve both the process and the product besides minimize the rework and the cost of correcting defects in further phases. Finally, we observed that the techniques can be applied even the SYSMOD process is not being used. In this case, it is enough that the pair of artifacts dialed by each technique is available.

Keywords: Embedded Systems; SysML; Simulink; Reading Technique; Verification and Validation; SYSMOD; Development Process.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BDD	Block Definition Diagram
Def.	Defeito
Diag.	Diagrama
Disc.	Discrepância
IBD	Internal Block Diagram
MEF	Máquina de Estados Finitos
REQ	Requirement Diagram
SE	Sistema Embarcado
SMD	Sate Machine Diagram
SML	Simulink
SYSMOD	System Modeling Process
$T1_{in}$	Subparte da Técnica T1 para o conceito de Input
$T1_{int}$	Subparte da Técnica T1 para o conceito de Interrupções
$T1_{mem}$	Subparte da Técnica T1 para o conceito de Memória
$T1_{out}$	Subparte da Técnica T1 para o conceito de Output
$T1_{reg}$	Subparte da Técnica T1 para o conceito de Registradores
$T2_{in}$	Subparte da Técnica T2 para o conceito de Input
$T2_{int}$	Subparte da Técnica T2 para o conceito de Interrupções
$T2_{mem}$	Subparte da Técnica T2 para o conceito de Memória
$T2_{out}$	Subparte da Técnica T2 para o conceito de Output
$T2_{reg}$	Subparte da Técnica T2 para o conceito de Registradores
$T3_{par}$	Técnica T3 para o conceito de Paralelismo
$T4_{comp}$	Técnica T4 para o conceito de Composição
$T5_{cond}$	Técnica T5 para o conceito de Condição

SUMÁRIO

Capítulo 1
INTRODUÇÃO

A área de Engenharia de Software procura promover avanços para o desenvolvimento de métodos, técnicas e estratégias que possam ser utilizados em diferentes áreas. Este capítulo trata de algumas dessas iniciativas e ressalta a importante lacuna de pesquisa tratada na proposição da tese deste trabalho.

1.1 Contexto

Um Sistema Embarcado (SE) refere-se a um sistema computacional de processamento de informações dedicado. Um SE não é visto isoladamente como um produto *standalone*, mas sim, como um elemento incorporado a um produto final como, por exemplo, robôs, carros, trens, aviões e equipamentos de telecomunicações, entre outros. Assim, pode-se pensar que um SE consiste em um conjunto de elementos (hardware e software), formado por dispositivos mecânicos, elétricos e um software (LIGGESMEYER; TRAPP, 2009 e MARWEDEL, 2011). Sabendo-se que hardware e software em um SE são dependentes, é razoável assumir então que SEs são sistemas computacionais elaborados para um propósito específico e são diferentes de outros sistemas computacionais como aqueles que são utilizados em computadores pessoais e supercomputadores (NOERGAARD, 2005, p. 5).

O processo de desenvolvimento de um SE envolve o conhecimento e uso de propriedades como: controle de temperatura, controle de iluminação, além de formas de uso típicas como portas lógicas e acionadores, registradores, interrupções, gerenciamento e controle de memória (EEPROM), gerenciamento e controle de entrada e saída através de dispositivos como portas lógicas, atuadores, motores e acelerômetros, e concorrência.

Na concepção de um SE, deve-se considerar, além dos requisitos diretamente ligados à funcionalidade da aplicação, o gerenciamento de requisitos não-funcionais como limitação de recursos e também a condição de ambiente extrema como calor, umidade, radiação, entre outros a que será submetido (LIGGESMEYER; TRAPP, 2009, p. 20).

É importante ressaltar ainda que fatores como: avanço tecnológico, área de aplicação e variabilidade das funcionalidades exigidas tem levado a uma mudança na forma como o processo de desenvolvimento de um SE é estabelecido.

Na literatura, verifica-se que há um aumento na demanda de elaboração de novas funcionalidades e uma produção de novos SEs em curto prazo (GRAAF et al., 2003 e LIGGESMEYER; TRAPP, 2009). Além disso, exige-se uma maior qualidade do SE, aliada com segurança e confiabilidade. Esse fato tem levado a comunidade acadêmica a explorar e estabelecer novos processos e técnicas de desenvolvimento de SEs e integrá-los com

diferentes áreas do conhecimento. Isso pode ser evidenciado pela crescente investigação sobre a redução dos riscos de falhas de um SE (KANDT, 2009 e KO; KANG, 1999).

De acordo com Ebert e Jones (2009) a prática de desenvolvimento utilizada em um SE é diferente das práticas aplicadas, por exemplo, na área de TI (Tecnologia da Informação) ou em aplicações para *desktops*. Para o domínio de SEs práticas mais formais e rigorosas aliadas a um processo de desenvolvimento são fundamentais para se garantir a qualidade do SE. Nesse contexto, estão inseridos também os sistemas de segurança crítica ou Sistemas Embarcados Críticos (SECs), que exigem que propriedades de qualidade como confiabilidade, desempenho, tolerância a falhas, temporização e agendamento de tarefas sejam verificados e validados nas etapas iniciais do processo de desenvolvimento (FEILER et al., 2004, p. 35, 2006, p. 5–6).

A complexidade inerente aos diferentes domínios de aplicações de SEs, aliada à necessidade de se detectar defeitos de forma rápida e precisa, são fatores importantes a serem considerados no desenvolvimento de SEs. Assim, os pesquisadores buscam critérios, ferramentas, métodos e processos que visam promover a qualidade do SE. Um grande avanço no estudo de SEs ocorreu com o surgimento da Engenharia de Sistemas Embarcados ─ área que alia os Sistemas Embarcados com a Engenharia de Software ─ na quais técnicas e estratégias tradicionais da Engenharia de Software têm sido são exploradas, a fim de se promover a qualidade no desenvolvimento do SE (GRAAF et al., 2003). Como consequência, pode-se observar o uso frequente de técnicas de modelagem com *Unified Modeling Language* (UML) e suas extensões como *Modeling and Analysis of Real-Time and Embedded Systems UML/RT* e UML/MARTE (OMG, 2010, 2011a, 2011b). Além disso, notam-se também tendências na elicitação e rastreabilidade de requisitos como aquelas abordadas na linguagem SysML (OMG, 2010), e práticas de desenvolvimento como MDA (*Model Driven Architecture*) (PASTOR; MOLINA, 2007).

Atualmente, artefatos da SysML/UML têm sido incorporados em processos de desenvolvimento de software, sobretudo para SEs, como é o caso do processo SYSMOD (*System Modeling*) (SYSMOD, 2013 e WEILKIENS, 2008). Esse processo vem se destacando devido à sua utilização prática pela comunidade de Sistemas Embarcados (CES); pelo uso frequente em projetos reais, como por exemplo, *Ativa Phase Experiment* (APE)[1] financiado pela Agencia Espacial Européia (ESO); e pela facilidade de uso de suas Diretrizes de desenvolvimento.

Essas abordagens descritas e outras têm sido exploradas no contexto de SEs por grupos de pesquisas nacionais e internacionais, como é o caso do IEEE-IFIPWG: 10.4 [2], CESAR[3] e o INCT-SEC[4].

O INCT-SEC (Instituto Nacional de Ciência e Tecnologia — Sistemas Embarcados Críticos, 2011) visa elevar o nível de conhecimento, competência e qualidade no desenvolvimento de sistemas embarcados críticos (SEC). Para alcançar esse objetivo, vem agregando habilidades, competências e infra-estrutura necessárias para o desenvolvimento de SECs. Além disso, pretende capacitar a academia e a indústria brasileira com o ensino, treinamento, pesquisa e desenvolvimento científico-tecnológicos em aplicações de relevância e de alto impacto econômico-social em áreas estratégicas do país. Dentre as aplicações desse tipo de sistema, podem-se destacar aplicações na agricultura, segurança e defesa nacional, aviação e meio ambiente.

Embora sejam verificadas importantes iniciativas que contribuem para a melhoria da qualidade no desenvolvimento de SEs, ainda é notável a carência por pesquisas envolvendo

[1] http://mbse.gfse.de/
[2] http://www.dependability.org/wg10.4/
[3] http://www.cesarproject.eu/
[4] http://www.inct-sec.org/

atividades de verificação e validação (V&V), como forma de garantia de qualidade dos diagramas.

Vale destacar também, que apesar da existência de técnicas e estratégias tradicionais de modelagem e de desenvolvimento de software, como previamente citadas, os desenvolvedores de SEs, em geral, iniciam o desenvolvimento de um SE a partir dos diagramas próximos do próprio código fonte (ANTONIO et al., 2011). Em um *survey* conduzido em colaboração com Freire (2011), pode-se observar essa tendência. Em particular, foi possível caracterizar o procedimento adotado por desenvolvedores de SEs no contexto do INCT em que se observou que 63% dos participantes, de um total de 19, relataram fazer uso de técnicas informais para a modelagem de SEs. Salienta-se que embora linguagens como LabVIEW (ANTONIO, 2008) e Simulink (KEHTARNAVAZ; GOPE, 2006 e TEWARI, 2002) sejam amplamente utilizadas pela comunidade de sistemas embarcados devido as suas facilidades de uso, nota-se ainda uma importante lacuna de pesquisa, sobretudo no que se refere às atividades de V&V de tais diagramas (ANTONIO et al., 2012; FREIRE et al., 2011).

1.2 Motivação e Objetivos

Considerando o contexto apresentado anteriormente, salientam-se os seguintes pontos:

- Complexidade inerente dos SEs;
- Objetivos relacionados ao projeto INCT-SEC, com destaque para a intenção de melhorar a competência e qualidade no desenvolvimento de SEs;
- Carência por Atividades de Verificação e Validação (V&V) em diagramas;
- Carência de um processo de desenvolvimento na área de SEs, permeado por atividades de Verificação e Validação (V&V), em particular, que utiliza diagramas SysML e Simulink; e
- Investigações mais recentes na área de SEs, na qual se notam várias iniciativas de uso de técnicas tradicionais de Engenharia de Software voltadas para SEs.

Assim, o objetivo deste trabalho é contribuir para melhorar a qualidade de artefatos que possam ser usados em um processo de desenvolvimento de SEs. Em particular, dá-se ênfase para diagramas SysML e Simulink, que são diagramas utilizados no processo SYSMOD. Essa contribuição está na definição de técnicas de leitura que levam em consideração elementos inovadores como o uso de normas internacionais de certificação de SEs como UL-98 e DO-178C (DANIELS, 2011 e HAYHURST et al., 2001), além de elementos pertinentes às estruturas das linguagens SysML e Simulink. As técnicas definidas ajudam na detecção de defeitos nesses diagramas, à medida que tais são construídos.

Portanto, a definição da tese deste trabalho de doutorado, pode ser enunciada da seguinte forma:

> *"Técnicas de Leitura baseadas em Normas de Certificação para Sistemas Embarcados e na estrutura das linguagens SysML e Simulink, são capazes de detectar e antecipar a identificação de defeitos em modelos baseados nessas linguagens, à medida que eles são construídos."*

1.3 Metodologia de Desenvolvimento do Trabalho

A metodologia utilizada para desenvolver a tese apresentada anteriormente consistiu nas seguintes etapas:

(1) Caracterização das principais lacunas de pesquisas existentes na área, por meio de seis Mapeamentos Sistemáticos (PETERSEN et al., 2008) planejados com o suporte da técnica "Goal Question Metric" (GQM) (BASILI et al., 1994; PRESSMAN, 2009, p. 617). Esse planejamento está apresentado na Figura 7.1 do **Apêndice A**, o GQM foi estabelecido da seguinte maneira:

- ***Goal*:** corresponde à tese proposta, no sentido de "Avaliar se a tese proposta é pertinente". Para fazer essa avaliação, esse objetivo foi desmembrado em seis outros objetivos específicos que correspondem a avaliar a existência ou não de trabalhos (publicações) em seis temas diretamente relacionados à tese.
- ***Question*:** corresponde às questões a serem investigadas para avaliar os objetivos específicos estabelecidos.
- ***Metric*:** corresponde às informações a serem coletadas nos trabalhos encontrados na literatura, para mapear os assuntos relacionados à tese proposta.

Uma vez elaborado o modelo GQM, preencheram-se os protocolos dos mapeamentos sistemáticos. A partir desses protocolos foram criadas strings de busca que foram aplicadas às máquinas de busca. Para gerenciar a execução dos mapeamentos foi utilizada a ferramenta StArt (ZAMBONI et al., 2010).

(2) Com base no mapeamento sistemático e considerando-se que o objetivo do trabalho era definir técnicas de leitura para dar suporte a atividade de inspeção dos diagramas que possam ser usados em um processo de desenvolvimento de SEs estabeleceu-se como alvo os seguintes itens: (i) as linguagens SysML e Simulink por serem amplamente usadas na especificação de SEs; (ii) o processo SYSMOD, por contemplar todas as principais etapas do desenvolvimento de um sistema e por fazer uso das linguagens citadas; (iii) as normas de certificação UL-98 e DO-178C, que são adotadas para certificar código de SEs, para que, com base nelas, se conseguisse definir um conjunto de técnicas de leitura que pudessem antecipar a detecção de defeitos que seriam detectados apenas quando o código fosse certificado.

(3) Considerando as escolhas feitas anteriormente, as técnicas de leitura foram especificadas com base em trabalhos similares, para contextos diferentes, encontrados na literatura (DENGER; CIOLKOWSKI, 2003 e MARUCCI et al., 2002 e TRAVASSOS; SHULL; CARVER; et al., 1999). Durante a construção dessas técnicas foram incorporados elementos importantes para a identificação de defeitos relativos, por exemplo, à concorrência, definidos por elementos sintáticos e semânticos existentes em alguns dos diagramas como os Diagramas de Requisitos, de Máquinas de Estados e de Sequência da SysML. Além disso, foram mapeados do nível de código para o nível de modelo alguns tipos de defeitos utilizados na certificação de código de SEs, pelas normas de certificação UL-98 e DO-178C (DANIELS, 2011). Para alcançar esse objetivo, concomitantes à definição das técnicas de leitura foram definidos os artefatos que seriam utilizados na atividade de inspeção, incluindo: Diagrama de Requisitos (REQ), Diagrama Interno de Blocos (IBD), Diagrama de Definição de Blocos (BDD), Diagrama de Estados (MEF), Diagrama de Sequências (SEQ) e Simulink.

(4) Concomitantemente à etapa (3) foram definidos e conduzidos estudos experimentais (experimentos e Exemplos de Aplicação) que permitiram avaliar gradativamente as técnicas de leitura propostas.

1.4 Organização do Trabalho

O Capítulo 1 apresentou o contexto de pesquisa envolvido nesse trabalho, em particular, com destaque para as principais fronteiras e desafios de pesquisas envolvidos na área de Engenharia de Software (ES) e de Sistemas Embarcados (SEs). Também no Capítulo 1 foi apresentada a proposição da tese de pesquisa que norteia este trabalho bem como a metodologia utilizada no decorrer deste trabalho de pesquisa.

No Capítulo 2 apresenta-se a revisão bibliográfica, em especial, estabelecida a partir das principais fontes de interesse (livros e artigos científicos), coletados a partir de um conjunto de Mapeamentos Sistemas realizados no curso deste trabalho.

No Capítulo 3 apresentam-se em detalhes os passos que foram empregados para a definição de uma família de técnicas de leitura. Além disso, no Capítulo 3 será apresentado o processo definido para a criação de uma família de técnicas de leitura bem como o processo SYSMOD permeado por essa família de técnicas de leitura.

No Capítulo 4 será apresentada a família de técnicas de leitura definidas a partir do processo inicialmente estabelecido no Capítulo 3.

No Capítulo 5 serão apresentados dois estudos experimentais aplicados em um ambiente controlado para avaliar a viabilidade de uso de uma das técnicas de leitura definidas no Capítulo 4.

No Capítulo 6 apresentam-se três Exemplos de aplicação que exploram a utilização da família de técnicas de leitura propostas. Assim, apresenta-se, passo a passo, a aplicação das técnicas em três exemplos, dos quais dois fizeram uso do processo SYSMOD e o outro, embora não tenha utilizado esse processo, foi desenvolvido com o suporte de artefatos da SysML.

Finalmente, no Capítulo 7 são apresentadas as Conclusões e em seguida os elementos pós-textuais (Referências e Apêndices) utilizados neste trabalho.

Capítulo 2
REVISÃO BIBLIOGRÁFICA: FUNDAMENTAÇÃO TEÓRICA

Atividades de Verificação e Validação aplicada a Sistemas Embarcados envolve importantes aspectos relacionados com a Engenharia de Software tradicional. Este capítulo apresenta uma revisão da literatura destacando as principais contribuições que serviram de inspiração para a definição da família de técnicas de leitura propostas no decorrer deste trabalho.

2.1 Considerações Iniciais

Este capítulo apresenta a fundamentação teórica necessária utilizada como referência e inspiração para a proposição de uma família de técnicas de leitura (denominadas neste trabalho de *Reading Technique for SysML and Simulink* - RTSS) apresentadas nos Capítulos 3 e 4 a seguir.

Assim, este capítulo esta organizado da seguinte maneira. Como primeiro ponto, será apresentado na Seção 2.2 uma visão geral de Sistemas Embarcados (SEs). Em seguida, na Seção 2.3 é destaca-se as técnicas de modelagens utilizadas pela comunidade de SEs em geral. Em seguida, na Seção 2.4 são apresentados conceitos e definições importantes sobre processos de desenvolvimento para SEs. Nessa seção, também são tratadas as principais normas de certificação utilizadas como garantia de qualidade de processos e produtos da área de SEs, como por exemplo, UL-98 e DO-178C. Finalmente, na Seção 2.5 são apresentados trabalhos tradicionais da área de Engenharia de Software, com destaque para um MS realizado que teve como objetivo avaliar e identificar possíveis lacunas de pesquisas envolvendo Atividades de V&V.

2.2 Sistemas Embarcados

Embora existam diferentes definições de SEs (Sistemas Embarcados), duas definições operacionais serão utilizadas no contexto deste trabalho. A primeira definição trata um SE do ponto de vista de um produto. Nesse contexto, um *Sistema Embarcado* é qualquer sistema computacional aplicado que está oculto (encapsulado) dentro de um produto (NOERGAARD,

2005 e SIMON, 1999). Os autores (EBERT; JONES, 2009)definem que um SE pode ser entendido do ponto de vista de um sistema de micro-controlador projetado e construído dentro de um equipamento técnico. Tais sistemas são construídos com uma específica finalidade e geralmente não é permitido carregar novas aplicações ou acrescentar novos periféricos. Essa característica inerente aos SEs, de não permitir o acréscimo de novas funcionalidades é o que estabelece a fronteira entre os SEs e os computadores de propósito geral(SIMON, 1999). Quando necessário, a comunicação com o ambiente ocorre via sensores e atuadores. Sistemas Embarcados podem fornecer uma interface para ações dedicadas. O software que executa em um sistema embarcado é conhecido como *Software Embarcado,* sendo parte integral do sistema.

A segunda definição utilizada diz respeito à abstração de um SE. Segundo (MARWEDEL, 2011, pp. 8–9), um *Sistema Embarcado* é tipicamente considerado como sendo um *sistema reativo*. Um s*istema reativo* é um sistema que possui um conjunto de estados e que continuamente interage com seu ambiente externo. Cada interação é executada a partir de uma entrada o que leva o sistema a realizar o processamento de alguma computação e a produzir uma saída, após o processamento, o estado inicial do sistema reativo pode ser alterado para um novo estado.

Ebert e Jones (2009) definem *Software Embarcado* como um *software* de *propósito-especial* construído em um sistema mais amplo. Em geral, há dois fatores que influenciam no desenvolvimento de novas aplicações dos SEs. O primeiro trata-se da ampla *demanda* por sistemas e *softwares* embarcados. Segundo Hatebur (2006), por exemplo, 98% das CPU's atualmente produzidas já utilizam algum tipo de sistema embarcado. O outro fator refere-se ao modo de operação de um SE, que é dito *crítico*, quando atua com certas restrições operacionais ou de ambiente como, por exemplo, um marca-passo[5]. Nesse cenário, cada vez mais novas tecnologias aliadas a processos rigorosos de controle de qualidade,são importantes no desenvolvimento de SEs.

Segundo Dubois et al.(2010), um SE possui um conjunto de requisitos funcionais e não-funcionais essenciais e que devem ser identificados em diferentes níveis de abstração ou visões. As principais visões segundo Dubois et al.(2010) são: visão do engenheiro de controle, visão do engenheiro de software e visão de integração. Essas visões são importantes, pois permitem (de diferentes perspectivas) a identificação de requisitos funcionais e não-funcionais essenciais para o funcionamento de um SE. Ainda de acordo com o autor, os principais requisitos não-funcionais que devem ser identificados em um SE são propriedades de tempo-real, características físicas de hardware, características físicas de software, desempenho, aspectos de variabilidade de plataforma (ambiente/sistema operacional) e restrições de segurança. O entendimento de um SE em termos de perspectivas também pode ser observado na arquitetura GENESYS (OBERMAISSER; KOPETZ, 2009, p. 45-48). Na estruturação de um SE quatro visões devem ser seguidas. A primeira visão trata do ambiente de execução e a troca de mensagens entre os componentes do SE (*Run-Time View*), a segunda visão trata do projeto do SE (*Design-Time View*), a terceira visão lida com o projeto físico e específico de plataforma (*Dependability View*) e finalmente a visão de consumo e controle de energia (*Power and Energy View*). Ressalta-se ainda que outras visões/perspectivas de SEs podem ser encontradas em outros trabalhos da área (BUESCHER; WILKINSON, 1990; DENGER; CIOLKOWSKI, 2003; MENKHAUS; ANDRICH, 2005a).

[5]Um *marca-passo* é um pequeno dispositivo que é colocado no peito ou abdômen para o controle da arritmia cardíaca pela variação da taxa de corrente elétrica em células *sinus node* ou *sino atrial* (SA) do coração - http://www.nhlbi.nih.gov/health/health-topics/topics/pace.

2.2.1 SysML

De acordo com a especificação da OMG (2010), a linguagem SysML é uma linguagem de modelagem de propósito-geral especialmente projetada para aplicações da área de engenharia de sistemas, incluindo por exemplo, Sistemas Embarcados e Sistemas Embarcados Críticos.

Em um evento internacional reunindo o consórcio (INCOSE) sobre *Model Driven Engineering* e representantes da OMG em 2001, foram estabelecidos os esforços necessários para padronizar as diferentes linguagens utilizadas por engenheiros de sistemas e adaptar a linguagem UML para o uso em aplicações de área de engenharia de sistemas. Desse esforço inicial entre a OMG e INCOSE em março de 2003 é liberada a primeira versão da proposta de construção de uma nova linguagem a partir da *core* da UML. A Linguagem de Modelagem SysML (ou brevemente denominada ao longo deste texto de SysML) prove suporte para a especificação, análise, projeto, verificação e validação de uma ampla faixa de sistemas complexos. Alguns exemplos de aplicações incluem a indústria automotiva, aeroespacial, comunicação, sistemas de informação (OMG, 2010 e WEILKIENS, 2008).

A SysML reutiliza um subconjunto de elementos da UML 2 e provê extensões adicionais. A Figura 2.1 ilustra a interseção entre a UML e também as extensões realizadas por meio do mecanismo de perfil da UML.

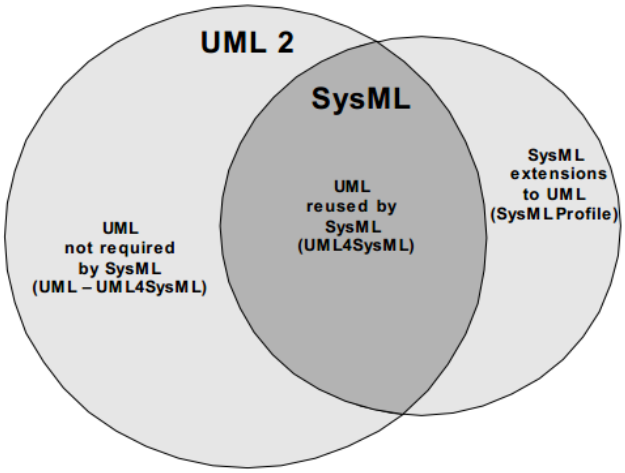


Figura 2.1 - Visão geral da SysML (OMG, 2010, p. 7).

Como principais aspectos de projeto da SysML, pode-se citar:

a) *Requirements-driven* — provê mecanismos para satisfazer a geração de artefatos a partir de requisitos previamente especificados em UML;
b) Reuso da UML — provê mecanismos para minimizar a carga de mudanças entre a linguagem UML e SysML por meio do reuso de elementos sintáticos e semânticos;
c) Extensão da UML — provê extensão padronizada dos elementos da SysML e UML; e
d) Interoperabilidade — provê mecanismos de intercâmbio de dados através de XMI.

Em resumo, para ilustrar os principais diagramas existentes no contexto da SysML a Figura 2.2apresenta a taxonomia entre os diagramas originais da linguagem UML, diagramas modificados da UML e novos diagramas desenvolvidos para a linguagem SysML.

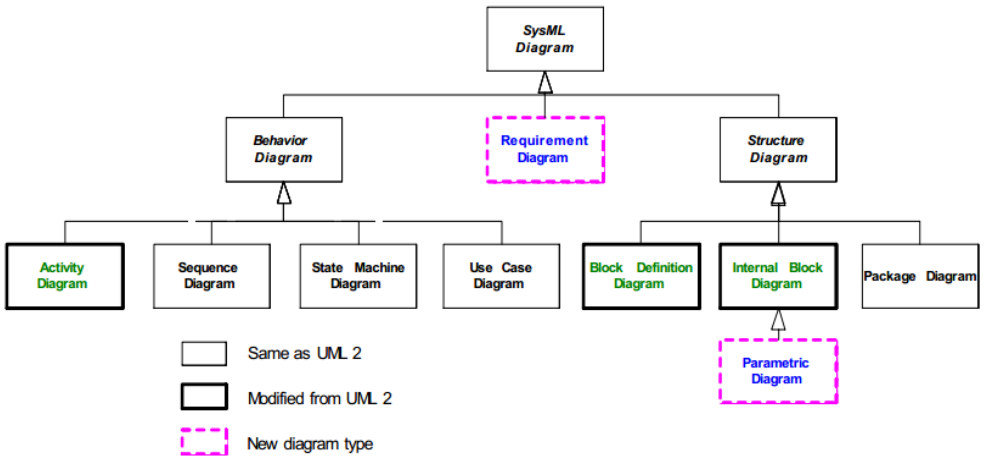


Figura 2.2 - Diagramas da SysML (OMG, 2010).

Como pode ser verificado na Figura 2.2, a SysML agrupa três categorias principais que são, Diagrama de Comportamento, Diagrama de Requisitos e Diagrama de Estrutura. Em cada categoria, a SysML faz o reuso dos diagramas da UML, alguns *estritamente*, ou seja, são totalmente reutilizados sem a inserção de modificações, como é o caso do Diagrama de Sequência (SEQ), Diagrama de Máquina de Estados (MEF), Diagrama de Casos de Uso e Diagrama de Pacotes. Por outro lado, os Diagramas de Atividades, Diagramas de Definição de Blocos (BDD) e Diagramas Internos de Blocos (IBD) são parcialmente modificados pelo mecanismo de extensão e *profile* da UML. Por exemplo, o Diagrama Interno de Blocos e Diagrama de Definição de Blocos são uma variação dos diagramas originais da UML, Diagrama de Estrutura Composta e Diagrama de Classes acrescidos do elemento de "Bloco".

2.2.2 Simulink

O MATLAB(MATHWORKS, 2012)é considerado uma linguagem para a computação de alto-nível e um ambiente para o desenvolvimento e simulação do comportamento dinâmico de sistemas embarcados. Em especial, o Simulink pode simular algoritmos, prover visualização de dados (2-D e 3-D) e realizar análise de dados e computação numérica. Com o MATLAB é possível fazer a integração com aplicações e linguagens de programação externas como C, C++, Fortran, Java, COM e Microsoft Excel. A partir do ambiente de desenvolvimento do MATLAB é possível o gerenciamento de código, arquivos e dados. Essa característica é importante para a leitura de dados externos a partir de uma fonte fixa, por exemplo, um arquivo que representa uma matriz com as coordenadas de um ponto no espaço *x*, *y*, *z*. Além disso, o MATLAB fornece um conjunto de bibliotecas para uma grande quantidade de aplicações, como análise e processamento de sinais, bioinformática e funções matemáticas (álgebra linear, estatística, análise de Fourier, filtros, otimização e integração numérica) (MATHWORKS, 2012).

O Simulink faz parte do ambiente de desenvolvimento do MATLAB, Simulink é um ambiente para simulação multi-domínio e se caracteriza por ser um ambiente *Model Based Design*(MBD)(MATHWORKS, 2012). Um ambiente MBD é definido como um conjunto de blocos predefinidos, o desenvolvedor pode construir diagramas dinâmicos (um modelo dinâmico é a representação rigorosa (matemática) de um sistema que evolui ao longo do tempo, um exemplo de um sistema dinâmico é o movimento de uma partícula em um campo

de potencial, que é descrito por meio de uma equação diferencial) com o intuito de se simular e verificar o comportamento do modelo construído.

Um exemplo de modelo Simulink é apresentado na Figura 2.3. O bloco *Sine Wave* gera uma onda senoidal a partir do tempo. O sinal que sai do bloco *Sine Wave* por conectores é bifurcado e passa para um bloco que realiza um cálculo integral. Após sair do bloco *Integrator*, o sinal é multiplexado (barra vertical preenchida) com o sinal que sai do bloco *Sine Wave*. O sinal multiplexado então é conectado a um osciloscópio.

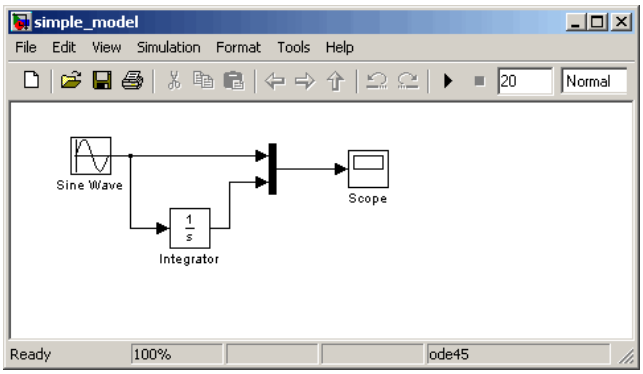


Figura 2.3 - Exemplo de um Modelo Simulink (MATHWORKS, 2012).

O modelo permite estabelecer os parâmetros de configuração de cada bloco, inclusive tempo de início e término da simulação; tipo de equação utilizada; importar/exportar dados de configuração e gerar o código C/C++ para ser executado no SE.

2.3 Processo de Desenvolvimento de Software

De acordo com Pressman (2009)o software é composto por três elementos principais, (1) programa de computador; (2) estrutura de dados e (3) documentação, sendo que cada elemento é criado como parte de um processo da Engenharia de Software. Segundo o IEEE (1990), a Engenharia de Software é definida como:

> *"Engenharia de Software: (1) é a aplicação sistemática, disciplinada bem como uma abordagem quantificável para o desenvolvimento, operação e manutenção de software; que é a aplicação da engenharia para o software. (2) O estudo das abordagens como em (1)"*
>
> (IEEE, 1990).

Assim, podemos ver a Engenharia de Software como um conjunto de atividades para a construção de *software* com alta qualidade.

Dentro das inúmeras atividades relacionadas com a Engenharia de Software, podemos citar a criação de técnicas, elaboração de processos, métodos de qualidade, métodos formais de prova, verificação, validação, teste de software, confiabilidade matemática dos diagramas, métodos de estimativas de índice de qualidade do *software* final (PRESMAN, 2009). Essas atividades estão relacionadas com as fases do ciclo do processo de *software*, que se inicia com a formulação dos requisitos, desenvolvimento de um produto de *software*, sua manutenção e finalização com sua renovação a partir de novas funcionalidades.

Um processo de desenvolvimento de *software* define-se como um *arcabouço* para as tarefas que são necessárias para se construir um *software* final com qualidade. Um processo define a forma para o controle e gerenciamento do projeto de *software*, bem como o contexto de aplicação dos artefatos (diagramas, documentos, dados, formulários, programas etc.) produzidos ao longo do processo (PRESMAN, 2009).

Ao longo do processo, prazos são estabelecidos, a qualidade é assegurada e a mudança inerente dos requisitos de *software* deve ser apropriadamente gerenciada. A fim de que se possa resolver um problema, o engenheiro de *software* ou a equipe de engenheiros deve adotar uma estratégia de desenvolvimento que compreenda os processos, métodos, e ferramentas necessárias para a evolução do processo. A estratégia utilizada no processo é conhecida como *modelo de processo* ou paradigma de engenharia de software (PRESMAN, 2009).

2.3.1 Mapeamento Sistemático sobre Processos de Desenvolvimento para Sistemas Embarcados

Essa seção procura destacar e caracterizar a área de Sistemas Embarcados por meio de um Mapeamento Sistemático (MS). Esse MS corresponde a um dos objetivos do modelo GQM apresentado Figura 7.1 do **Apêndice A**.

A intenção desses MS em relação à tese proposta (Capítulo 1) foi de avaliar a existência de processos de desenvolvimento para SÉS, pois havendo tais processos pretendia-se estudá-los para identificar seus pontos positivos e negativos, a fim de levá-los em consideração nesta proposta de tese. Além disso, se houver na literatura um processo de desenvolvimento de SEs que contemple todas as características aqui pretendidas, a proposta de tese teria que ser revisitada. Por outro lado, não se encontrando estudos que abordem processos de desenvolvimento para SEs, haveria indícios de que o processo a ser considerado neste trabalho se traduz em uma proposta inovadora e que pode contribuir de modo promissor para a área. Considerando a tese proposta no Capítulo 1 bem como o MS realizado os seguintes pontos podem ser sintetizados: (i) não foram encontrados relatos na literatura de processos de desenvolvimento de software que envolvam atividades de verificação e validação, sobretudo, com o uso de artefatos SysML e elementos pertinentes a certificação, concorrência e representação hierárquica de tais artefatos; (ii) não foram encontrados estudos relacionados que relatem processos de desenvolvimento de software permeados por técnicas de leitura para artefatos da linguagem SysML; (iii) embora não envolvam atividades de verificação e validação, foram encontrados processos de desenvolvimento, já bem conhecidos pela comunidade de SEs e que usam artefatos da SysML e alguns casos elementos de certificação.

Os principais resultados encontrados por meio desse MS serão apresentados a seguir, a partir de uma discussão sobre os estudos identificados via o MS (vide Figura 2.24[6]).

Em seguida, na Seção 2.4.2 será apresentada uma síntese do processo de desenvolvimento para sistemas embarcados *System Modelling Process* (SYSMOD) bem como as justificativas pertinentes que levaram ao uso de tal processo como inspiração para este trabalho.

[6] Gráfico de bolha construído com a ferramenta MS Excel

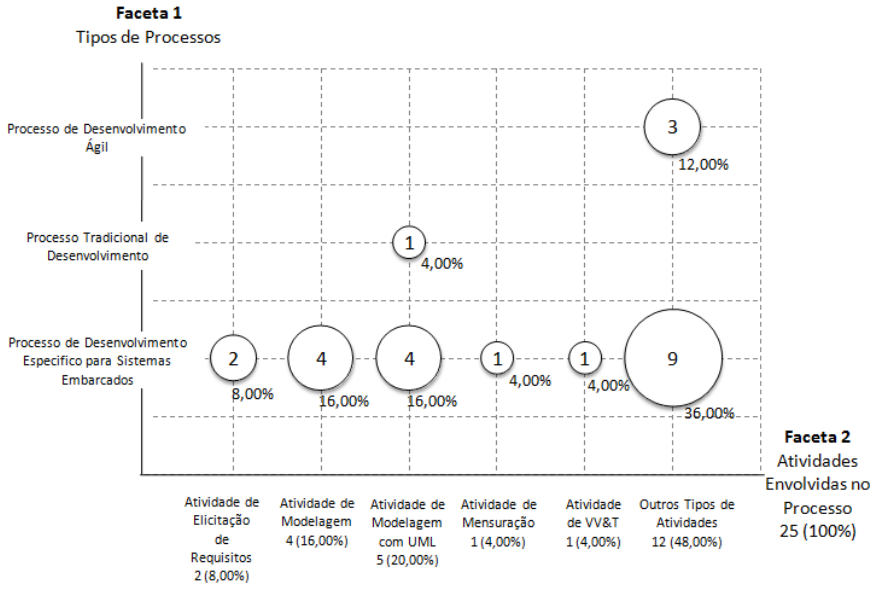


Figura 2.4 - Gráfico de Bolhas para os tipos de processos e atividades envolvidas no processo em relação às facetas (1) Tipos de Processos; e (2) Atividades envolvidas no processo.

Com o intuito de extrair as informações adequadas para contemplar o objetivo deste MS, foram estabelecidas duas facetas: (1) *Tipos de Processos* pois nessa faceta foram agrupados os estudos em categorias que envolviam processo de desenvolvimento ágil, processo de desenvolvimento tradicional e processo de desenvolvimento específico para SEs; e (2) *Atividades envolvidas no processo*, pois nessa faceta foram criadas categorias para agrupar atividades de elicitação de requisitos, atividades de modelagem, atividades de modelagem com UML, atividades de mensuração, atividades de verificação e validação e outros tipos de atividades. O agrupamento e classificação nas facetas (1) e (2) foi realizado à medida que os resumos dos estudos selecionados eram lidos. A leitura dos resumos permitiu obter informações relevantes acerca da investigação que estava sendo realizada, as quais foram, posteriormente, agrupadas em categorias associadas com o tema das duas facetas estabelecidas inicialmente. No total, foram coletados 48 estudos e de acordo com o processo de MS foram selecionados 21 estudos para a leitura do texto completo.

Para facilitar o entendimento do leitor será feita uma breve descrição com respeito à leitura do gráfico de bolhas, ressaltando uma leitura vertical dos estudos associados com a faceta (2), pois o maior interesse de investigação nesse MS eram as atividades envolvidas no processo. A faceta (1) contém o grupo de categorias associadas com tipos de processos. Os rótulos de cada uma dessas categorias foram elaborados a partir do agrupamento dos estudos semelhantes e também com base no formulário de seleção de dados do MS.

Como exemplo de leitura no gráfico de bolhas, observe que com base no MS definido 20% dos estudos exploram o uso de atividades de modelagem com UML, dos quais apenas um estudo se aplica processos tradicionais de desenvolvimento, os outros estudos são específicos para o contexto de SEs.

Portanto, a partir do gráfico da Figura 2.4, foi possível explorar os principais grupos de estudos selecionados por meio do MS retratando as principais atividades envolvidas no processo. Em seguida, será apresentada uma breve descrição dos artigos desse MS mais relevantes. Dos estudos selecionados, 20% relacionam algum tipo de atividade de modelagem com UML no processo de desenvolvimento para SEs. Nessa linha de estudos, os autores (HEISEL; HATEBUR, 2005), apresentam um processo de desenvolvimento para SEs que é decorrente de experiências práticas na indústria. Esse processo conhecido como DePES (*De-*

velopment Process for Embedded Systems), é uma abordagem adaptativa dividida em 12 etapas bem definidas. A abordagem proposta pelos autores possui duas variações DePES-I e DePES-II. O processo de desenvolvimento DePES-II é o atual e tem sofrido constantes alterações por ser um processo intimamente ligado com a indústria. O processo DePES-II inicia com o levantamento de requisitos com base na abordagem *problem frames*(JACKSON, 2001). Em seguida, uma proposta de arquitetura em camadas é utilizada, elementos e componentes de software são então especificados utilizando Máquinas de Estados, para que possam ser aplicadas de teste em diagramas(HATEBUR, 2006 e HEISEL; HATEBUR, 2005). O processo DePES-II apresenta características importantes que devem ser analisadas e podem ser incorporadas e/ou adaptadas para o contexto deste trabalho no que se refere ao modo de divisão em etapas e também como são feitos a elicitação de requisitos e a representação dos interesses funcionais e não-funcionais que são descritos através da abordagem de *problem frames*.

Com relação ao processo de desenvolvimento ágil Savolainen et al., (2010) tratam da necessidade das organizações em acelerar o processo de desenvolvimento melhorando a habilidade das empresas em reagir de forma rápida às mudanças dos requisitos. Os autores mostram a aplicação de processo de desenvolvimento ágil em SEs e afirmam que técnicas ágeis e métodos tradicionais de desenvolvimento de software devem ser combinados especialmente em dois casos: quando existe uma grande diferença de conhecimento (*skills*) dos desenvolvedores e quando a empresa precisa distribuir os requisitos para um grupo grande de desenvolvedores. Com base nesse trabalho foi possível identificar o interesse da comunidade em se retratar um processo leve e ágil que pode ser útil para a elaboração do processo de desenvolvimento a ser proposto neste trabalho. Além disso, o modo como os diferentes tipos de artefatos são representados ao longo do processo pode influenciar na efetiva concepção do software embarcado. Para evitar ambigüidades na descrição dos artefatos, será investigado como representar a funcionalidade dos artefatos ao longo do processo com o apoio de Diretrizes (*guidelines*), o que pode no final promover a qualidade dos artefatos e também de todo o processo de desenvolvimento.

De acordo com os trabalhos de Post et al. (2008), uma importante atividade aliada ao processo de desenvolvimento é o uso de atividades de teste aplicadas a diagramas (*Model-Checking*). Os autores propõem que para reduzir o tempo necessário para a construção de um software embarcado utilizando processos tradicionais de desenvolvimento devem-se utilizar meios para checagem automática de defeitos. O processo de desenvolvimento tradicional em cascata é dividido em duas grandes etapas, a primeira consiste em uma interpretação abstrata de um modelo do sistema embarcado seguida da validação combinada do modelo abstrato e o código em linguagem C. Os trabalhos de Post et al. (2008), fornecem subsídios teóricos essenciais para indicar que o uso dos diagramas de mais alto nível de abstração antes da implementação caracteriza uma essencial etapa do processo de desenvolvimento que pretende-se desenvolver neste trabalho. Aliado a isso, o uso de atividades de verificação e validação ao longo do processo pode fornecer importantes evidências dos defeitos antes mesmo de o software embarcado ser colocado em produção. Outros trabalhos como de (MELLEGAARD; STARON, 2010), também mostram diferentes tipos dos diagramas aplicados - de mais alto nível de abstração até o mais baixo nível de abstração -, que são aplicados ao longo do processo de desenvolvimento de SE. Embora os autores, explorem o uso de artefatos de mais alto nível, ao longo do processo de desenvolvimento, os autores não apresentam as etapas e as Diretrizes de transformação entre os diagramas. Tais estudos, portanto, fornecem fortes indicadores da lacuna de pesquisa atualmente e da contribuição do processo de desenvolvimento que pretende ser desenvolvimento neste trabalho de doutorado.

Ainda em relação a estudos sobre transformação dos diagramas e Diretrizes de mapeamento, foram identificados ao longo do MS estudos aplicando a técnica de modelagem UML

para a área de SEs. A partir da Figura 2.5, pode-se observar que 36% dos estudos (a soma da categoria *Atividades de modelagem* e *Atividades de modelagem com UML*) envolvem algum tipo de uso dos diagramas na definição ou execução do processo. A partir desse comportamento, foi observado durante a leitura dos estudos que existe uma crescente tendência no uso dos diagramas para a representação de problemas complexos a fim de que se possa reduzir o tempo, entendimento e o custo de desenvolvimento. Essa abordagem tem empregado o uso de técnicas MDE (*Model-Driven Engineering*), de forma que o desenvolver possa modelar um sistema complexo e como conseqüência, utilizando geradores automáticos, produzir um código para uma plataforma específica de domínio (LIGGESMEYER; TRAPP, 2009).

As propostas SARAH (*Software Architecture Reliability Analysis Approach*) (TEKINERDOGAN et al., 2008) e FMEA (*Failure Modes and Effects Analysis*) (MENKHAUS; ANDRICH, 2005 e PENTTI; ATTE, 2002) são processos de desenvolvimentos específicos para o contexto de SEs, em tais processo a característica importante são as etapas de elaboração de arquitetura e também detecção de falhas que são elaboradas nas etapas iniciais do processo do desenvolvimento.

Outra contribuição relevante e que deve servir de inspiração para o processo a ser desenvolvimento neste trabalho é o trabalho de Meira (2005). Ressaltam a importância de atividades de verificação como uma atividade essencial do ciclo de vida do software. Aliado a isso é fundamental que se tenham esforços na área para que se possa melhorar a eficácia de atividades de verificação e validação e como conseqüência a qualidade do software embarcado. De acordo com o autor, as técnicas utilizadas para a validação do projeto de SEs é categorizada em: simulação e técnicas formais. Uma limitação associada com a simulação de SEs é devido à complexidade inerente de projeto dos SEs, o que pode tornar a validação exaustiva impraticável. Embora o autor não defina um processo completo para a área de SE,o autor propõe uma metodologia *top-down* para validação de SEs utilizando uma abordagem combinada de técnicas formais e simulação. Além disso, é possível verificar no trabalho a existência de um comparativo entre as abordagens *botton-up* e *top-down* de desenvolvimento, que pode ser útil para futura comparação e avaliação do processo de desenvolvimento a ser proposto neste trabalho.

Uma perspectiva importante para a definição de um processo de software é o uso de certificação. De acordo com a certificação DO-178B/ED-12B (HAYHURST et al., 2001) o propósito do processo de verificação é detectar e reportar erros que foram introduzidos no processo de desenvolvimento. A certificação DO-178B foi estabelecida pela FAA[7] (*Federal Aviation Administration*) tem o intuito de se assegurar a aprovação de aspectos de software em aeronaves e os requisitos de equipamentos de aero-navegabilidade. Essa certificação também é amplamente utilizada para a certificação e garantia de confiabilidade dos softwares embarcados desenvolvidos pela agencia espacial americana (NASA). Em relação ao processo de software, ressalta-se que a certificação DO-178B prioriza o uso de atividades de verificação como parte integral de um processo de desenvolvimento. Dessa forma, incluindo atividades de verificação em cada atividade de desenvolvimento pretendida, isso pode ajudar a promover a qualidade em cada fase (etapa) de desenvolvimento. Uma evidência importante decorrente da leitura da certificação é o destaque para identificação de defeitos já nas primeiras fases de desenvolvimento priorizando a qualidade, de acordo com a norma, a aplicação de atividades de teste nas etapas finais do ciclo de desenvolvimento do software é impraticável.

A Figura 2.5 sumariza as principais atividades envolvidas no processo segundo a certificação DO-178B. Na figura é possível verificar que a verificação é parte integral do

[7] http://www.faa.gov/

processo de desenvolvimento, sendo, portanto uma atividade que deve ser planejada alimentando assim todo o ciclo do processo de desenvolvimento de software. Com base no MS realizado verificou-se que a certificação DO-178B tem sido utilizada na maioria dos estudos que envolvem SEs com características críticas, pretende-se a partir do levantamento realizado investigar a aplicação da certificação DO-178B no contexto do processo a ser desenvolvimento, em particular visando o uso de Simulink como um software elaborado a partir do processo de desenvolvimento a ser proposto.

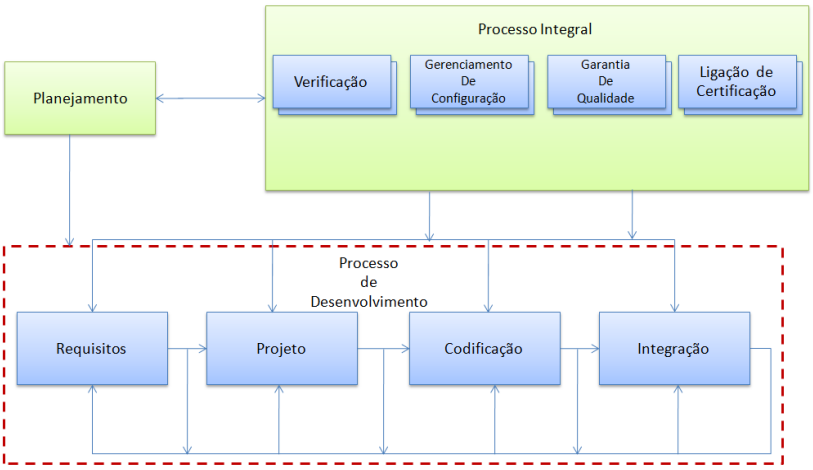


Figura 2.5 - Processo de desenvolvimento segundo norma de certificação DO-178B/ED-12B (adaptado(HAYHURST et al., 2001)).

Publicada originalmente em 1992, a norma de certificação de requisitos de *aeronavegabilidade* e aspectos de software aéreos DO-178B/ED-12B possui as seguintes características (DANIELS, 2011): (a) Representa um consenso da comunidade de aviação desde 1992; (b) Independe to tipo do processo de software utilizado; (b) Define cinco níveis de software definidos como software mais crítico (*Level A*) até software menos crítico (*Level E*); (c) Ênfase em atividades de teste baseada em requisitos; (d) Análise de cobertura estrutural; (e) Exige rastreabilidade para assegurar que todos os requisitos de teste de cobertura estejam de acordo com o código fonte; e (e) Evitar realizar suposições *quantitativas* sobre a confiabilidade do software. De acordo com Daniels (2011) o uso de uma norma de certificação como a DO-17B/ED-12B não implica que o software desenvolvido está totalmente livre de defeitos (isso está muito distante de ser o caso), no entanto, o uso de uma norma de certificação auxilia na validação dos requisitos determinando que estes estejam em conformidade com o código elaborado. Ressalta-se também, que um dos grandes desafios da comunidade de aviação em geral, é a evidencia de dados públicos que mostrem a relação entre acidentes causados por falhas relacionadas a software, hardware e as certificações existentes, como por exemplo, DO-178B. Nesse contexto, Daniels (2011) traz um importante estudo, identificando possíveis categorias e suas relações com quinze acidentes aéreos que parecem estar relacionados com a norma de certificação DO-178B/ED-12B.

Tendo em vista a importância de normas de certificação aplicadas no contexto de aeronaves, em Dezembro de 2004 o comitê RTCA e EUROCAE decidiu realizar atualizações na norma DO-178B/ED12B para refletir muitos dos aspectos que a Comunidade de Engenharia de Software tem realizado desde 1992. Em particular, a base da norma original foi mantida, adicionando-se elementos para adequação de aspectos formais, orientação a objetos e projeto baseado em diagramas (*Model-Based Design*). A partir desses termos de referencia ficou estabelecido as Diretrizes para a atualização da norma para DO-178C/ED12C (DANIELS, 2011). Atualmente, software com suporte a *Model-Based Design*, como o

MATLAB/Simulink, já inclui aspectos de certificação para as normas DO-178B e DO-178C (MATHWORKS, 2013).

Além da norma de certificação DO-178B/C, outra norma de uso geral para SEs é a norma UL-98 (DESAI, 1998 e UNDERWRITERS LABORATORIES, 1998).

A UL-98 é uma norma de certificação internacional ampla e de propósito geral. Seu objetivo é avaliar e certificar diferentes tipos de equipamentos e componentes vinculados a plataformas de um sistema embarcado. Dentre os principais componentes avaliados e certificados pela UL-98 pode-se citar: motores cardíacos, refrigeradores, *frezers*, ar condicionados, utensílios de cozinha, aquecedores de água, utensílios de uso pessoal como barbeadores elétricos, dispositivos de computação em geral tais como PCs e computadores de bordo. Como se pode observar, a UL-98 cobre uma ampla gama de SEs. Vale destacar, que assim como a norma de certificação DO-178C, a norma de certificação UL-98 apresenta um conjunto mínimo de requisitos de cobertura que devem ser atendidos para que um componente embarcado possa ser certificado.

Em suma, dado o propósito para a realização deste MS o que se pode concluir é que existem estudos associados com a elicitação de requisitos, atividades de modelagem - com e sem o uso de UML, atividades específicas de avaliação dos diagramas e atividades de verificação e validação. Uma evidência importante desse MS é que existe um grupo elevado de estudos (48%) reportando o desenvolvimento de SEs a partir de um estágio de mais baixo nível, em geral, da própria fase de programação, mas na maioria deles observou-se que os autores recorreram a algum outro tipo de artefato para poder explicar a aplicação. Isso leva a pensar que a necessidade de representar a aplicação de uma forma que não apenas em código, é uma necessidade geral nesse contexto.

Também a partir desse MS realizado, foi possível constatar evidencias de uso de normas de certificação internacional para o contexto de SEs. Em particular, inicialmente identificou-se a norma DO-178B/C que serviu de inspiração e Diretriz para a proposição das técnicas RTSS. A partir dessa norma, explorou-se outra norma de certificação internacional, a UL-98 por se tratar de uma norma de propósito geral e de ampla utilidade prática entre os desenvolvedores e também da comunidade de SEs.

Outro ponto de interesse nesse MS, e que vale salientar, é que foi possível identificar as principais lacunas existentes da área, em especial, neste trabalho será utilizado o processo de desenvolvimento SYSMOD, que tem sido amplamente utilizado pela comunidade de Sistemas Embarcados. Ressalta-se também que o processo SYSMOD, define um conjunto de *profiles* extensíveis e bem aceitos pela Comunidade de Engenharia de Sistemas (CESs) e que são amplamente aceitos como parte do processo de modelagem em ferramentas e plataformas de modelagem dirigidas a diagramas, por exemplo, “Enterprise Architecture” e “MagicDraw” (MAGICDRAW, 2013 e SPARX, 2013 e WEILKIENS, 2008). Uma lista popular das principais ferramentas de modelagem que fornecem suporte a modelagem SysML pode ser encontrada no endereço http://www.system-modeling.com/. A seguir será apresentada uma visão geral do processo SYSMOD bem como as respectivas justificativas de uso do processo SYSMOD no contexto deste trabalho.

2.3.2 SYSMOD

O objetivo desta seção é apresentar as principais características existentes no processo de desenvolvimento SYSMOD (WEILKIENS, 2008). O SYSMOD é um processo de desenvolvimento *top-down* e utiliza os principais diagramas da linguagem de modelagem SysML (SYSMOD, 2013). Dentre as principais características do SYSMOD destacam-se as atividades, os papeis bem como o produto gerado pelo processo de desenvolvimento SYSMOD. Para ilustrar isso, a Figura 2.6 retrata o *meta-modelo* utilizado no SYSMOD

considerando a sua composição e instanciação. Nessa figura, é possível verificar que existe um elemento "Produto" que represente o que será gerado como artefato. O elemento "Activity" retrata qual é a atividade utilizada para atingir o objetivo final estabelecido no produto. O elemento "Process Module", contemplam um conjunto específico de produtos, atividades e papeis relevantes a uma área específica.

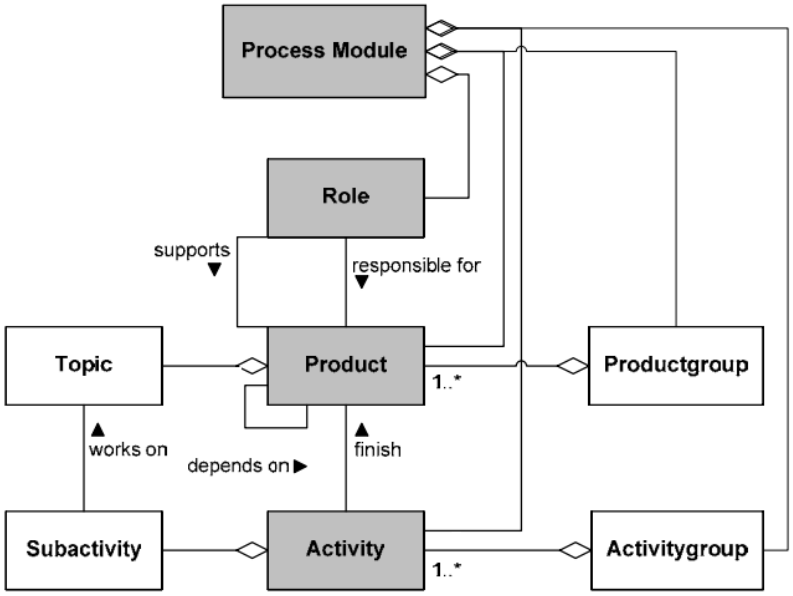


Figura 2.6 – Meta-modelo utilizado no SYSMOD (RAUSCH et al., 2005).

O processo SYSMOD (vide Figura 2.27) inclui um conjunto de cinco fases de desenvolvimento denominadas de "Requisitos", "Contexto do Sistema", "Casos de Uso", "Conhecimento de Domínio" e "Estrutura do Sistema". Cada uma dessas fases está inserida como uma abstração da análise e projeto do sistema sendo projetado. De acordo com a especificação do processo de desenvolvimento SYSMOD, as seguintes fases são resumidamente apresentadas a seguir:

Requisitos - envolve a coleta e sumarização dos requisitos em termos de requisitos essenciais e técnicos. Os requisitos essenciais descrevem o sistema independentemente da sua solução técnica, promovendo, assim o reuso de requisitos de mais alto nível de abstração e a aplicação destes requisitos para diferentes tipos de soluções específicas;

Contexto do Sistema - define as fronteiras do sistema embarcado que será projetado. O contexto do sistema, leva em consideração, as fronteiras do sistema bem como os principais atores envolvidos. Nessa etapa, são realizados sucessivos refinamentos, dos diagramas BDD e IBD. Desse modo, em cada passo de refinamento, o projetista consegue adicionar elementos aos diagramas como fronteiras do sistema, descrição dos fluxos com os respectivos atores do sistema, pontos de interação entre os atores do sistema com sistema embarcado que está sendo modelado. No último passo desta etapa, o projetista do sistema embarcado, consegue visualizar as especificações de interfaces do sistema em um nível de granularidade, mas claro para a especificação do projeto do sistema embarcado. De forma complementar, o processo do sistema pode ser projetado, para isso, o projetista, deve junto com os casos de uso, descrever de forma geral o fluxo global do sistema em termos de um diagrama de atividades. Esse passo é especialmente útil, pois permite identificar o fluxo da aplicação com alto nível de granularidade, sendo assim, está etapa promove a elicitação dos fluxos essenciais envolvidos no sistema;

Casos de Uso - envolve a especificação do sistema via casos de uso. Nesse momento, o projetista, deve estabelecer os aspectos importantes entre os requisitos e o sistema elaborando casos de uso essenciais. Originalmente, no processo de desenvolvimento SYSMOD, o projetista do sistema embarcado deve realizar a remoção de redundância nos casos de uso, identificando fluxos e casos de uso equivalentes durante a modelagem do sistema embarcado, o que pode em certos casos, ocasionar sérios problemas no projeto final se a consistência for violada. Por exemplo, em um sistema de computador de bordo, um caso de uso de uso que represente *adicionar rota*, e outro que retrate *modificar rota*, identificam interesses semelhantes, portanto, o uso de um glossário, para a identificação de similaridade e o respectivo processo de reorganização (do inglês, *refactorying*) são importantes etapas na fase de casos de uso. Também nessa fase de casos de uso, o projetista deve, após elaborar um caso de uso essencial, elaborar um caso de uso detalhado, descrevendo os aspectos de fluxos do sistema incluindo todas as suas variações e exceções existentes. Para elaborar um caso de uso detalhado, como recomendação o projetista deve desenvolver um diagrama de atividades descrevendo os fluxos associados sobre o ponto de vista dos casos de uso;

Conhecimento de Domínio - envolve a especificação do SE via diagrama de blocos (BDD). Nessa etapa, são especificados os principais blocos e seus respectivos relacionamentos fornecendo uma ampla visão da estrutura do sistema embarcado. Ainda nesta etapa, deve ser elaborado um diagrama de blocos utilizando BDD, identificando as principais interfaces operacionais do sistema que está sendo projetado. Nesse contexto, o sistema deve prever os contratos e restrições que devem ser definidas e especificadas. Ainda nessa etapa, o projetista deve levar em consideração os aspectos de restrição temporal, de *clock* e também de hardware fundamentalmente importantes para um sistema embarcado. Tais aspectos estão intimamente relacionados com os requisitos não funcionais de um sistema embarcado e devem ser descritos em termos dos diagramas de diagramas paramétricos;

Estrutura do Sistema - define o domínio da aplicação via um diagrama de sequências identificando aspectos e interesses comportamentais do sistema. Nesse momento, o projetista deve lidar com questões como, por exemplo, qual o tipo evento dispara uma transição no sistema embarcado. Também nesse momento, o projetista deve elaborar um diagrama de máquina de estados fornecerem uma abstração de mais alto nível do sistema em relação à visão comportamental do sistema.

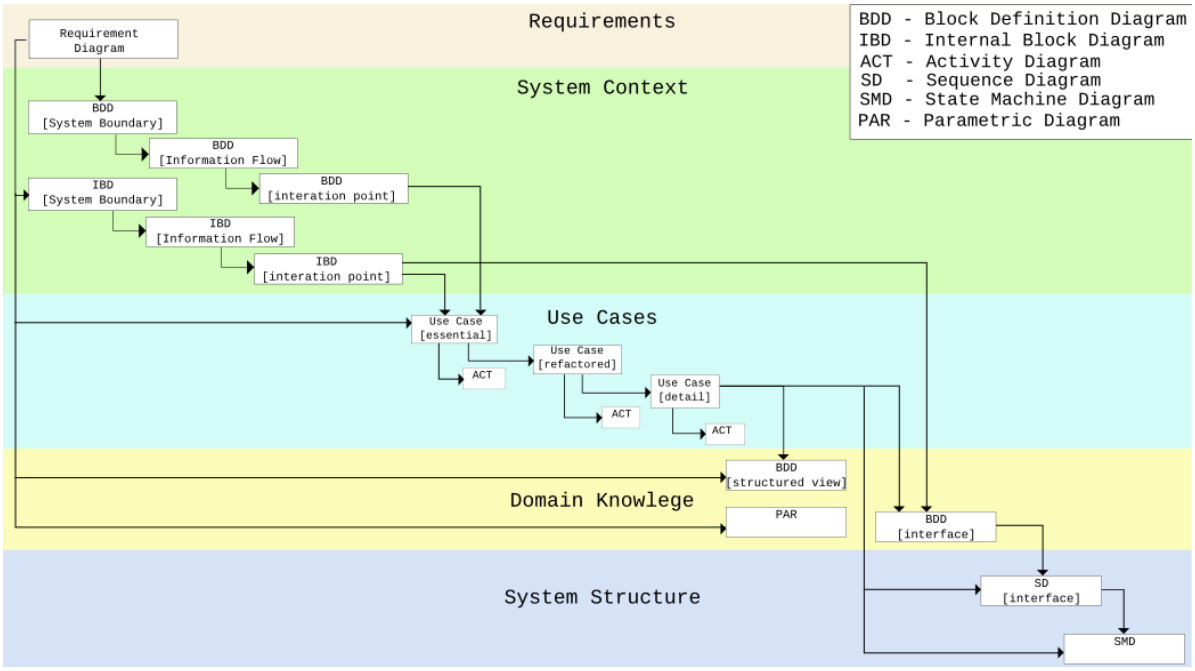


Figura 2.7 - Processo SYSMOD adaptado de (WEILKIENS, 2008).

Em suma, o SYSMOD é um processo de modelagem de sistemas genérico que inclui etapas e atividades bem definidas para a especificação top-down em SysML. Ressalta-se que o SYSMOD inclui aspectos de rastreabilidade por meio do uso de anotações diversas nos modelos. Como restrições, o processo SYSMOD não cobre aspectos particulares e específicos de projetos envolvendo empresas de domínios específicos. Desse modo, nem todo artefato produzido pelo processo SYSMOD deve ser elaborado da mesma forma, considerando diferentes tipos de áreas e domínios. Além disso, o processo de desenvolvimento SYSMOD não contempla aspectos específicos de Garantia de Qualidade (QA – *Quality Assurence*) como, por exemplo, Atividades de Verificação e Validação e Atividades de Mensuração de Software, essenciais para se garantir a qualidade dos artefatos que estão sendo gerados ao longo do processo de desenvolvimento. De modo geral, o processo SYSMOD, por retratar aspectos genéricos de desenvolvimento de software, aliado ao uso da linguagem de modelagem SysML, torna-se importante fonte de inspiração para o desenvolvimento deste trabalho. Nesse sentido, o SYSMOD provê uma estrutura e um conjunto de Diretrizes que podem ser estendidas, adicionando-se pontos importantes de Validação e Verificação e meios para se realizar a medida através de indicadores de qualidade ao longo do processo. Pensando nisso, um conjunto de atividades de garantia de qualidade foram propostas, e são relatadas no Capítulo 3, incluindo atividades de verificação e validação e métricas para avaliação da compreensão dos diagramas Simulink são incorporadas como parte das atividades de desenvolvimento do processo SYSMOD.

2.4 Atividades de Verificação e Validação (V&V)

As atividades de V&V (Verificação e Validação) garantem a qualidade do *software* e devem ser aplicadas durante todo o ciclo de desenvolvimento, com o intuito de aumentar a confiabilidade de forma que o produto final esteja livre de defeitos. As Atividades de

Verificação e Validação podem ser divididas em dois grupos conhecidos da área da Comunidade de Engenharia de Software: *Atividades estáticas* e *Atividades dinâmicas*. A inspeção é considerada uma atividade estática, pois ela não implica na execução do artefato (modelo) que está sendo inspecionado. Ela é baseada em técnicas de leituras que ajudam o inspetor a ler o artefato e buscar o maior número de defeitos possível. Assim, as técnicas de leitura são definidas para um determinado artefato sendo diferente para um documento de requisitos ou para o código de um programa. Além disso, se baseiam em geral, em uma taxonomia de defeitos e procuram dar apoio a uma leitura do documento inspecionado com base nos defeitos especificados nessa taxonomia.

As **técnicas de leitura** podem ser: informais, semi-formais e formais. As técnicas informais – *ad hoc* – se baseiam na experiência do inspetor e, portanto, não tem como serem reproduzidas ou melhoradas já que não se conhece o processo adotado pelo inspetor.

As semi-formais, como o ***checklist***, consistem em uma lista de perguntas direcionadas a encontrar os defeitos da taxonomia, conforme as perguntas que compõem o *checklist*. Embora seja baseada em um *checklist*, não há como se determinar o processo adotado pelo inspetor para manuseá-la e dessa forma apenas parte dela pode ser repetida.

As técnicas formais, em geral, são reproduzíveis, pois o inspetor segue um processo de leitura mais rígido. No caso de código fonte, há a técnica *Stepwise Abstraction*, que força o inspetor a ler as estruturas mais internas e depois as mais externas (MYERS; SANDLER, 2004; PRESSMAN, 2009).

Em se tratando de técnicas de leitura tradicionais, na literatura encontram-se importantes iniciativas que servem de inspiração para a proposição de novas técnicas de leitura (BASILI; CALDIERA; et al., 1996 e BASILI; GREEN; et al., 1996 e MARUCCI et al., 2002 e SHULL, 1998 e TRAVASSOS et al., 2000 e TRAVASSOS; SHULL; FREDERICKS; et al., 1999).

Como exemplo, podem-se citar as seguintes técnicas de leitura (i) **PBR - *Perspective Based Reading*** (BASILI; GREEN; et al., 1996) que é utilizada para inspecionar documentos de requisitos; (ii) **UBR - *Use Based Reading***, que é utilizada para detectar anomalias em interfaces de usuário (ZHANG, BASILI e SHNEIDERMAN, 1998); (iii) **OORTs - Object *Oriented Reading Techniques***(TRAVASSOS et al., 2000) que é utilizada para inspecionar diagramas UML no nível de projeto; (iv) **OORTs/ProDES** (MARUCCI et al., 2002) que é utilizada para inspecionar diagramas UML que são construídos de acordo com o processo ProDES; (v) ***Scenario Based Reading*** - também conhecida como ***Defect Based Reading***é uma técnica de leitura que explora defeitos em relação ao documento de requisitos (AURUM et al., 2002); e (vi) **TUCCA/GUCCRA**- é uma técnica de leitura que tem como objetivo que tem como objetivo principal auxiliar na construção dos diagramas de Casos de Uso, fornecendo procedimentos mais sistemáticos que direcionem a modelagem e permitam que o modelo construído não apresente tanta variação se desenvolvido por diferentes pessoas. À medida que o modelo é construído, essa técnica também propicia condições de inspecionar o Documento de Requisitos(BELGAMO et al., 2005)**.**

Para sintetizar as diferentes técnicas de leitura existentes os autores Basili et al. (1996) sugerem o uso de uma árvore para representar a relação entre o espaço do problema (que envolve os diferentes tipos caminhos associados com a tecnologia de técnicas de leitura) e o espaço de solução (que são as soluções específicas direcionados pelas técnicas de leitura propriamente). A Figura 2.28 sumarizar as principais técnicas de leitura mencionadas anteriormente com auxílio dessa árvore. Essa figura é uma adaptação do original do autor Basili et al. (1996), proposta por Belgamo (2005).

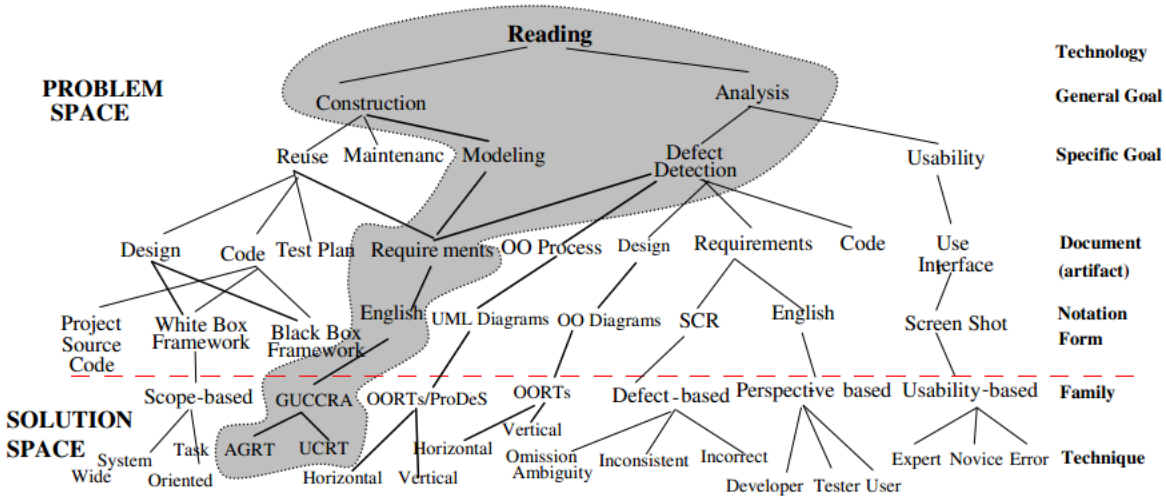


Figura 2.8 - Família de Técnicas de Leitura adaptado de Belgamo (2005, p. 37).

Nessa árvore (Figura 2.28), pode-se observar, por exemplo, em destaque as técnicas GUCCRA que envolvem no primeiro ramo da árvore (em objetivo geral), tanto aspectos para a análise e identificação de defeitos, quanto para a construção de artefatos envolvidos na atividade de inspeção.

Portanto, Atividades de V&V, sobretudo técnicas de leitura, são investigadas a partir de diferentes pontos de vistas. No entanto, ainda é carente na área de Sistemas Embarcados e em Engenharia de Software, trabalhos que explorem atividades de inspeção para diagramas SysML, que são amplamente utilizados pela comunidade de sistemas embarcados.

Neste trabalho, embora não explorado em profundidade, existem ao contrário da atividade de inspeção, o atividades dinâmicas para identificação de defeitos, tipicamente aplicadas em código fonte. Nesse contexto, a atividade de teste é caracterizada por executar o artefato que está sendo avaliado. As principais técnicas de teste são: Funcional, Estrutural e Baseada em Defeitos. Elas são complementares já que cada uma é mais propícia a encontrar determinados tipos de defeitos.

A **técnica Funcional ou Teste Caixa Preta**, tem por objetivo encontrar discrepâncias entre o programa e as especificações externas. Uma especificação externa é uma descrição do comportamento do programa do ponto de vista do usuário (requisitos do sistema) (MYERS; SANDLER, 2004). Exemplos de critérios de teste funcionais são: particionamento de equivalência, análise do valor limite, grafo de causa efeito e erro por adivinhação (*error guessing*).

A **técnica Estrutural ou caixa-branca** tem o objetivo de avaliar a estrutura interna de um programa, conduzindo para a validação da estrutura lógica interna de um programa. Essa técnica permite avaliar características internas de um programa tais como: controle de fluxo, fluxos condicionais e desvios. Exemplos de critérios de teste funcionais são: todos os nós e todos os arcos (MYERS; SANDLER, 2004).

A **técnica Baseada em Defeitos** tem por objetivo os defeitos que são cometidos com maior frequência pela equipe de desenvolvimento. Exemplos de Critérios são: análise de mutantes e semeadura de erros (MYERS; SANDLER, 2004).

2.4.1 Mapeamento Sistemático sobre Atividades de Verificação e Validação para Sistemas Embarcados

Essa seção procura destacar e caracterizar a área de Sistemas Embarcados através de um Mapeamento Sistemático identificando as principais contribuições recentes sobre atividades de verificação e validação. Esse mapeamento refere-se a um dos objetivos do modelo GQM apresentado na Figura 7.1do **Apêndice A**.

A intenção desse MS em relação à tese proposta foi de explorar a existência de elementos tais como técnicas de Verificação e Validação (V&V) para SEs, pois o uso de tais técnicas (estratégias e processos) de V&V podem promover a qualidade do processo de desenvolvimento de software. Como consequência, caso fossem encontrados estudos que indicassem o uso desses elementos de V&V, então tais estudos deveriam ser considerados na elaboração da tese proposta. Por outro lado, caso não fossem encontrados estudos explorando esses elementos de V&V, e considerando a relevância dessas atividades para a qualidade de software, isso caracterizaria a importância e inovação desta proposta.

A Figura 2.29resume os dados desse MS.

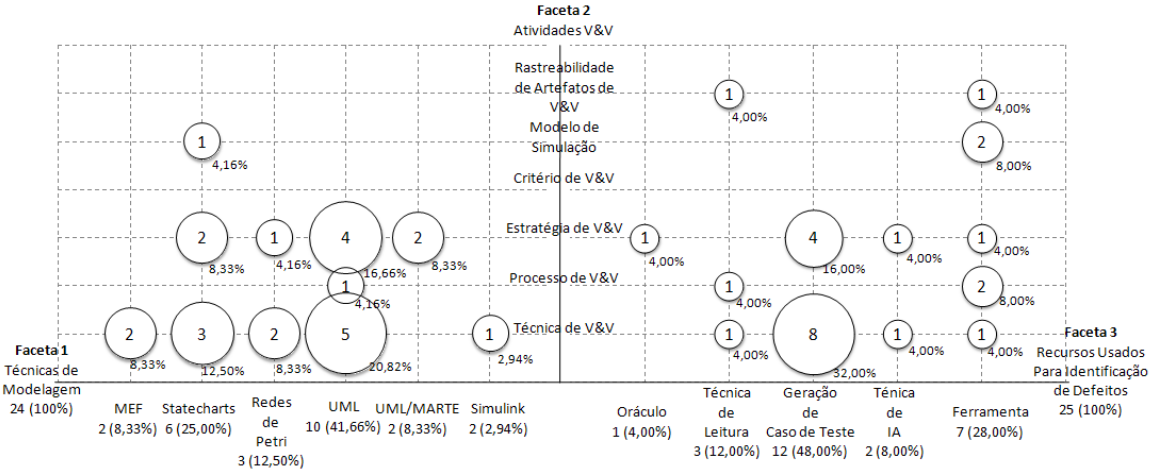


Figura 2.9 - Gráfico de Bolhas para processos, técnicas, estratégias e critérios de SE, agrupando as facetas (1) Técnicas de modelagem; (2) Atividades de V&V; e (3) Recursos usados para identificação de defeitos.

Com o intuito de identificar os elementos descritos anteriormente, foram estabelecidas três facetas*:* (1) *Técnicas de modelagem,* foram agrupados os estudos com técnicas de modelagem formais e semi-formais como MEFs, statechart, Redes de Petri, UML, UML/MARTE e Simulink, (2)*Atividades de V&V* com relatos do uso de atividades de verificação e validação e (3) *Recursos usados para identificação de defeitos,* como oráculo, técnica de leitura, geração de casos de teste, técnica de IA e ferramenta para SEs. O agrupamento e classificação nas facetas (1), (2) e (3) foi realizado à medida que os resumos selecionados eram lidos. A leitura dos resumos permitiu obter informações relevantes acerca da investigação que estava sendo realizada, que foram posteriormente, agrupadas em categorias associadas com o tema de cada faceta estabelecida. No total, foram coletados 411 estudos e de acordo com o processo de MS foram selecionados 80 para a leitura completa. No gráfico de bolhas são apresentados os estudos mais significados, 24 estudos na faceta (1) e 25 estudos na faceta (2) perfazendo um total de 49 estudos. Para facilitar o entendimento do leitor foi feita uma breve descrição da leitura do gráfico de bolhas, com destaque para uma leitura vertical dos estudos das facetas (1) e (3), pois o maior interesse de investigação nesse MS eram as

técnicas de modelagem e os recursos utilizados para a identificação de defeitos. Os rótulos da faceta (2) para cada categoria foram elaborados a partir do agrupamento de estudos semelhantes e com base no formulário de seleção de dados do MS.

Como exemplo de leitura do gráfico de bolhas (Figura 2.9), pode-se notar que há um estudo que referência a linguagem Simulink e nesse mesmo estudo também são abordadas técnicas de V&V. Por outro lado, não foram encontrados estudos relatando o uso de Simulink como alvo de atividades de V&V. Ainda é possível verificar que existe um grande interesse na área de SEs quanto ao uso de recursos para a geração de casos de testes atingindo um total de 48% dos estudos selecionados. Ressalta-se ainda que 25% dos estudos referenciam o uso de statechart como técnica rigorosa para a modelagem de SEs. Além disso, nesse conjunto de estudos envolvendo statechart, percebeu-se a existência de investigações sobre o uso de técnicas e estratégias de V&V para statechart e ainda a simulação dos diagramas statechart combatividades de V&V.

Com a análise do gráfico (Figura 2.9) ainda é possível explorar os principais grupos de estudos selecionados com o apoio do MS, retratando a técnica de modelagem, atividades de V&V e os recursos utilizados para a identificação de defeitos. A seguir foram selecionados alguns estudos mais relevantes para esse MS. Podemos dar um destaque especial ao interesse no uso ou aplicação de técnicas combinadas de verificação e validação (V&V) envolvendo técnicas de modelagem como, por exemplo, UML ou UML/MARTE perfazendo aproximadamente 50% dos estudos selecionados. De acordo com (KO; KANG, 1999), o uso de atividades de verificação em projetos de SEs são determinantes para detecção de falhas antes mesmo de o software ser embarcado em hardware, prevendo assim futuras falhas operacionais. Essas falhas podem ser detectadas por estratégias e técnicas de V&V quando combinadas com artefatos de modelagem utilizados em estágios anteriores à codificação. Nos trabalhos de (ARCURI ET al., 2010)está apresentada a aplicação de uma abordagem *caixa-preta* para a atividade de teste. Os autores modelam o RTES (*Real-Time Embedded Systems*) com UML e o perfil MARTE para a produção de casos de testes para o RTES. É importante lembrar, que esses trabalhos trazem uma grande contribuição para o processo de desenvolvimento que está sendo proposto neste trabalho, pois mostram a importância do uso dos diagramas de mais alto nível de abstração e Diretrizes para a geração de casos de teste a partir dos diagramas de objetos desenvolvidos com o perfil UML/MARTE.

Ainda, a partir da Figura 2.9, também foi possível constatar que dos estudos que selecionados existe um interesse em atividades de teste em diagramas tais como *Model Based Testing* (MBT) e *Model Checking*. Nos estudos selecionados, pode-se constatar que a técnica MBT tem sido tratada em conjunto com UML, UML/MARTE e também com statechart (KO; KANG, 1999; SINHA; SMIDTS, 2006). As abordagens *Model-Based Testing* e *Model Checking* estão ligeiramente relacionadas, pois em ambos os casos é usado algum tipo de modelo (*model*). Segundo(SABHARWAL et al., 2010) a atividade *Model-Based Testing* é uma técnica caixa-preta para geração de casos de teste. A atividade MBT tem sido utilizada visando reduzir o tempo de aplicação da atividade de teste. Esse tipo de estratégia é emprega no contexto de sistemas embarcados a partir dos diagramas UML, SDL, Z, diagramas de estados, diagramas de fluxos de dados, diagramas de fluxos de controle, entre outros, para a produção de casos de testes e cenários de testes. Por outro lado, a atividade *Model-Checking* apresenta algum tipo de ferramenta (*Model-Checker*) que emprega a conversão de um modelo para uma especificação rigorosa por meio de um modelo matemático. Após essa conversão, esse modelo é avaliado aplicando-se uma prova de teoremas (KO; KANG, 1999).

De acordo com Cu et al. (2011), o desenvolvimento de sistemas de controle, principalmente aéreo, necessitam de atividades de Verificação e Validação (V&V). O trabalho evidencia a importância de *Model- Based Testing* como a técnica utilizada em todos os sistemas de controle aéreo. Ainda, identifica a definição de uma nova abordagem para

assegurar a confiabilidade dos diagramas elaborados em Simulink. Na abordagem tradicional *Model Based Testing*, descrita no trabalho, um modelo executável pode ser comparado com um código executável durante a fase de projeto do processo de desenvolvimento. O procedimento consiste em inserir a mesma entrada (caso de teste) para um modelo (por exemplo, Simulink) e para o código executável. A partir da saída obtida (do modelo Simulink e código executável), pode-se obter subsídios para indicar a correção da implementação (código executável). Uma condição necessária para a aplicação dessa estratégia é que o caso de teste deve cobrir todos os blocos no modelo, bem como todas as linhas de código e condições no código executável. Como consequência, o uso de outra fonte de comparação independente do modelo original utilizado, pode ser uma tarefa de difícil aplicação. Assim, no trabalho de Cu et al. (2011) pode-se perceber uma importante contribuição em termos do processo a ser desenvolvido em razão do procedimento utilizado para descrever a técnica MBT. Tal técnica deve ser considera como fonte inspiradora para a elaboração do processo de desenvolvimento pretendido com esse trabalho.

Os trabalhos de Arcuri et al. (2010) aparecem no gráfico da Figura 2.9como 8% dos estudos selecionados para técnicas de Inteligência Artificial. Nesses estudos, os autores apresentam técnicas como *Random Testing (baseline)*, *Adaptive Random Testing*, e *Search-Based Testing*. Neste último caso os autores usam algoritmos genéticos para realizar busca por casos e cenários de testes. No final os autores também apresentam um processo em que e é apresentando como utilizar os três tipos de técnicas de forma combinada.

Em relação às ferramentas de testes, cerca de sete estudos (28%) apresentam algum tipo de ferramenta. Uppal é uma ferramenta de *Model-Checking* que pode gerar casos de testes (HESSEL et al., 2008).Nos trabalhos de (ELAMKULAM et al., 2007), os autores mostram experiências práticas no uso das ferramentas IBM *Rational Rose RealTime* (RoseRT), que tem sido largamente usada para o desenvolvimento de sistemas reativos concorrentes e sistemas embarcados e dá suporte à UML statechart. IBM RuleBase é uma efetiva ferramenta para *Model-Checking*, na proposta dos autores, uma ferramenta converte automaticamente diagramas do RoseRT como entradas para o RuleBase. Embora o foco do processo desenvolvimento que está sendo proposto não seja a elaboração de uma ferramenta de conversão dos diagramas, é essencial identificar os elementos essenciais para conversão entre diagramas já existentes na área de SE bem como os principais diagramas de mais alto nível atualmente utilizados.

É possível verificar que transformação entre diagramas (fonte e alvo) é uma tarefa que carece de corretude no momento da transformação entre os diagramas. Essa transformação pode introduzir um defeito no modelo transformado (alvo). Assim, uma maneira de se assegurar que o modelo transformado (alvo) reflete o modelo inicial (fonte) é assegurar que certas propriedades do modelo fonte são preservadas no modelo alvo. Outra forma de se assegurar que os diagramas transformados apresentam corretude é expressar o modelo inicial como um meta-modelo e a partir desse meta-modelo produzir um conjunto de casos de testes (diagramas de testes e/ou parâmetros de entrada de testes) que devem ser aplicados no modelo final a fim de que se possa avaliar a sua cobertura (KARSAI; NARAYANAN, 2007 e SAMPATH et al., 2007).

No artigo intitulado “High Quality statechart through Tailored, Perspective-Based Inspection” (DENGER; CIOLKOWSKI, 2003), os autores apresentam um estudo de grande importância para a proposta de tese deste trabalho. No artigo, os autores, destacam o *statechart* como uma técnica fundamental para descrever o comportamento de sistemas dinâmicos. Além disso, os autores definem uma taxonomia de defeitos que indica oito critérios de qualidade devem ser cumprido por uma especificação de software que é baseada em *statechart*. Os principais critérios de qualidade descritos no artigo estão associados com o padrão IEEE 830 e são utilizados em *statechart* para a identificação de defeitos. Associada a

essa taxonomia de defeitos, os autores descrevem uma abordagem para inspeção em *statechart* combinando e adaptando Técnicas de Leituras PBR (*Perspective-Based Reading*) para uma especificação em *statechart*. A Técnica de Leitura apresentada faz parte de um projeto mais amplo denominado QUASAR[8] que procura lidar com os desafios da engenharia de requisitos e da garantia de qualidade do processo de desenvolvimento de SEs. Embora os autores apresentem uma abordagem para a inspeção e detecção de defeitos em *statechart* utilizando PBR, eles não relatam um estudo de caso, nem uma avaliação empírica controlada da técnica utilizada. Contudo, o artigo traz uma importante contribuição na maneira como combinar e descreveram as Diretrizes do *checklist* para a inspeção em *statechart* com base na taxonomia de defeitos. Portanto, o artigo deve ser considerado como um estudo essencial para a definição do processo que se pretende desenvolvedor neste trabalho.

Outra perspectiva de atividade de V&V no contexto de SE, pode ser observada na arquitetura de referencia GENESYS (OBERMAISSER; KOPETZ, 2009). Uma das atividades essenciais para se garantir a segurança e confiabilidade da arquitetura de referência é a atividade de teste. Na arquitetura de referencia, a atividade de teste é parte integral do ciclo de desenvolvimento do sistema o que pode impactar em custos globais associados com o desenvolvimento do sistema. A fim de reduzir o impacto com as atividades de teste no desenvolvimento do sistema, uma recomendação é a elaboração de uma arquitetura com interfaces padronizadas e bem definidas para os componentes. As interfaces padronizadas e bem definidas podem auxiliar a reduzir o impacto com as atividades de testes envolvidas, metodologias utilizadas para a integração de componentes e também treinamento técnico dos engenheiros de sistemas. Uma segunda recomendação para se reduzir o impacto global dos testes no sistema, é a elaboração dos requisitos ao longo do ciclo de desenvolvimento. Embora tenha sido possível identificar algumas recomendações para aplicação de atividades de teste, a arquitetura de referência GENESYS carece de um conjunto de diretivas mais precisas para a aplicação de atividades de Verificação e Validação (V&V) ao longo da elaboração da arquitetura. Desse modo, pretende-se investigar mais atentamente a definição do ciclo de desenvolvimento da arquitetura GENESYS, com o intuito de se determinar possíveis lacunas no que se refere a atividades de V&V. Tais lacunas são importantes evidências já que se pretende desenvolver um processo de desenvolvimento de software permeado por atividades de verificação e validação. Portanto, a arquitetura GENESYS pode ser uma fonte inspiradora para a presente proposta de doutorado. Por exemplo, embora exista uma recomendação sobre o uso da norma de certificação DO-178B (HAYHURST et al., 2001), não foi possível identificar uma relação entre o uso da norma e as respectivas Diretrizes para a elaboração das atividades de V&V já nas etapas preliminares de desenvolvimento do sistema através da arquitetura GENESYS.

Em suma, dado o propósito para a realização desse MS o que se pode concluir é que existe uma preocupação na área em se retratar o SE com auxílio de técnicas de modelagem de mais alto nível de abstração como é o caso de statechart, UML e UML/MARTE. Pode-se observar também a partir do MS, que o uso dos diagramas de mais alto nível de abstração está associado com estratégias e técnicas de verificação e validação. Embora não tenham sido encontrados estudos que combinem a linguagem Simulink e diagramas de mais alto nível de abstração como parte de um processo de desenvolvimento (conforme observação possível pelo MS da Seção anterior), essa evidência caracteriza uma importante lacuna de pesquisa atual na área de SEs. Essa lacuna é ainda mais significativa no contexto deste trabalho, uma vez que se o Simulink não está sendo tratado no contexto de um processo de desenvolvimento, um processo de desenvolvimento permeado de atividades de V&V é ainda mais relevante para melhorar a qualidade desse tipo de aplicação.

[8] Mais informações podem ser encontradas em http://publica.fraunhofer.de/

2.5 Considerações Finais

Nesta seção foram apresentados os principais pontos envolvendo a fundamentação teórica básica e revisão da literatura realizada através de um Mapeamento Sistemático. Nesse contexto, destacam-se os seguintes pontos apresentados: (i) processos de desenvolvimento de software para SEs - com destaque para o processo SYSMOD; e (ii) atividades de Verificação e Validação (V&V). Ressalta-se que a revisão da literatura aqui apresentada, proporcionou a identificação de lacunas de pesquisa importantes, como a carência na área de SEs, de um processo de desenvolvimento de software permeado por atividades de garantia de qualidade, como o uso de Atividade de Verificação e Validação (V&V). Outro ponto a ser ressaltado, é que embora o processo de desenvolvimento SYSMOD contemple Diretrizes para o desenvolvimento de SEs utilizando uma abordagem *top-down*, o mesmo não fornece atividades formais para a garantia da qualidade, como o uso de técnicas de leitura.

Considerando-se as lacunas de pesquisa apresentadas anteriormente, e aliado ao fato de não terem sido identificadas na literatura técnicas de leitura para diagramas SysML sobretudo para a identificação de discrepâncias relativas a certificação de SEs tais como UL-98 e DO-178C bem como o seu uso ao longo de um processo de desenvolvimento de software, será apresentado no Capítulo 3 e no Capítulo 4, um conjunto de técnicas de leitura definidas para fornecer garantir de qualidade para SEs. Além disso, também será apresentado como estas técnicas podem ser utilizadas como garantia da qualidade ao longo do processo de desenvolvimento SYSMOD.

Capítulo 3
METODOLOGIA DE DEFINIÇÃO DAS TÉCNICAS DE LEITURA PARA SISTEMAS EMBARCADOS

A elaboração de técnicas de leitura pode ser uma atividade desafiadora, em especial, para a área de Sistemas Embarcados. Este capítulo apresenta um processo para a definição da família de técnicas de leitura (RTSS) e as Diretrizes de uso dessas técnicas ao longo do processo SYSMOD.

3.1 Considerações Iniciais

No Capítulo 1 foram apresentados os objetivos deste trabalho bem como a proposição da tese que é a definição de técnicas de leitura para diagramas SysML e Simulink, as quais têm o propósito de apoiar à detecção de defeitos nesses diagramas.

No Capítulo 2 apresentou-se uma revisão da literatura sobre a área de SEs e as principais definições utilizadas no contexto deste trabalho.

Assim, com base nos Capítulos 1 e 2, foi possível estabelecer alguns pontos de fundamental importância para serem tratados durante a definição das técnicas ***Reading Techniques for SysML and Simulink*** **(RTSS)**. O primeiro deles é a importância no uso de um processo de desenvolvimento que possa proporcionar uma melhor qualidade no SE desenvolvido (WELKIENS, 2008; MBSE, 2005). Dessa forma, identificou-se na literatura o processo SYSMOD que, conforme explicado anteriormente, por estar baseado em diagramas especialmente em diagramas das linguagens SysML e Simulink, mostrou-se como uma alternativa viável, uma vez que esses tipos de artefatos são amplamente utilizados para a especificação de SEs. O segundo ponto que inspirou a definição das técnicas RTSS foi à questão da certificação de SEs, uma vez que quando visto sob o aspecto industrial, os SEs precisam ser certificados. Isso levou à adoção das normas de certificação UL-98 e DO-178C na definição das técnicas de leitura, com a intenção de anteciparem-se defeitos que são investigados no nível de código para o

nível dos diagramas, como, por exemplo, defeitos relacionados a interrupções, controladores de CPUs, memórias, entradas/saídas e estruturas de decisão. O terceiro ponto refere-se ao uso de uma taxonomia de defeitos previamente definida, assim a taxonomia IEEE STD1044:2009 foi utilizada na definição das técnicas visto que essa taxonomia envolve o uso de termos comuns para apoiar atividades de V&V. Finalmente, um quarto ponto que influenciou na definição das técnicas foi a questão de possíveis erros de transcrição entre diagramas, ao se passar de uma fase de desenvolvimento para outra subsequente. Salienta-se que, neste último caso, tomando por base as notações utilizadas nos diagramas SysML, exploraram-se possibilidades de defeitos de hierarquia e concorrência, visto que esses elementos estão associados às características críticas dos SEs.

Considerando-se o exposto anteriormente a árvore para representar as técnicas de leitura, inicialmente mencionada no Capítulo 2, foi atualizada para incorporar a família de técnicas de leitura RTSS e elementos associados ao contexto de Sistemas Embarcados (SEs). Assim, a Figura 3.1 ilustra as atualizações pertinentes com destaque para o objetivo que é a identificação de defeitos ("Defect Detection"), e que inclui dois novos nós: "ES Process"(i.e., do inglês, "Embedded System Process") e "Actor-Oriented" (que é o paradigma empregado em linguagens como, por exemplo, Simulink e/ou LabVIEW). Além disso, as formas de notação Simulink e SysML também foram incorporadas uma vez que as técnicas RTSS usam tais artefatos como entrada para as técnicas.

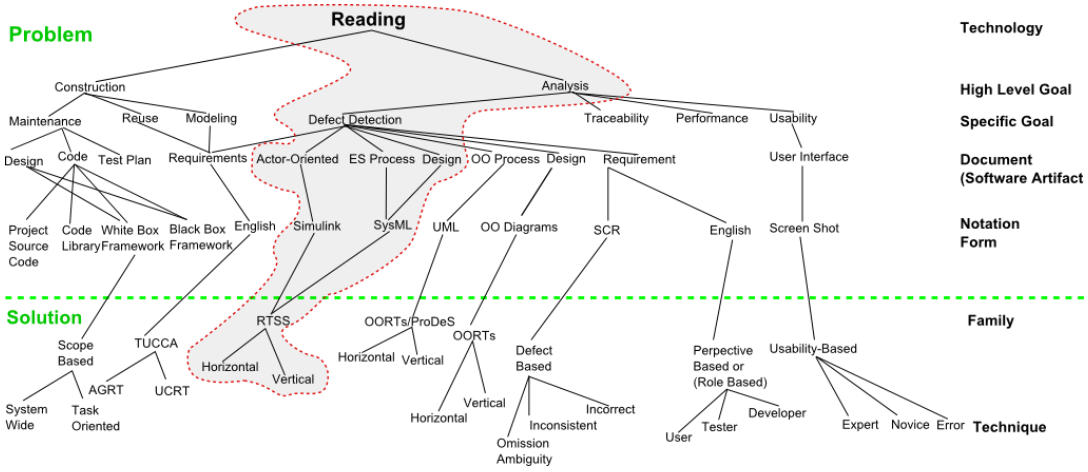


Figura 3.1 - Família de Técnicas de Leitura com a representação das técnicas RTSS (adaptado Belgamo (2005, p.37)).

Dessa forma, este capítulo sintetiza os passos que foram empregados para a definição de uma família de técnicas de leitura RTSS.

Este capítulo está organizado da seguinte forma. Na Seção 3.2 será apresentado o processo definido para a criação de uma família de técnicas de leitura. Na Seção 3.3 será apresentado o processo SYSMOD permeado pelas técnicas de leitura T1 a T5. Na Seção 3.4 são apresentadas as considerações finais.

3.2 Processo para a Definição as Técnicas RTSS

A definição das técnicas RTSS foi feita com base em um processo extraído da definição inicial da técnica T4. Essa técnica foi a primeira a ser elaborada, pois ela detecta defeitos em pares de diagramas (nesse caso, Diagrama de Requisitos x Diagrama de Estados).

Com o aprendizado adquirido na definição da T4 e tendo-se que considerar os elementos mencionados na Seção 3.1, elaborou-se o processo apresentado na Figura 3.2 para nortear e padronizar a definição das demais técnicas. Tal processo é sistematizado com auxílio da notação de diagramas de atividades da UML. Como pode ser observado, tal processo é composto de três etapas que se repetem para definir cada uma das técnicas de leitura (Etapas B, C e D) e de uma etapa preliminar que deve ser realizada uma única vez (Etapa A). Tais etapas são: Etapa A: essa etapa é de organização das fontes de defeitos que serão usadas para estabelecer os defeitos a serem investigados pela técnica; Etapa B - Seleção das fontes de defeitos que serão investigados pela técnica a ser definida; Etapa C - Modelagem da técnica de leitura; e Etapa D - Especificação rigorosa da técnica de leitura.

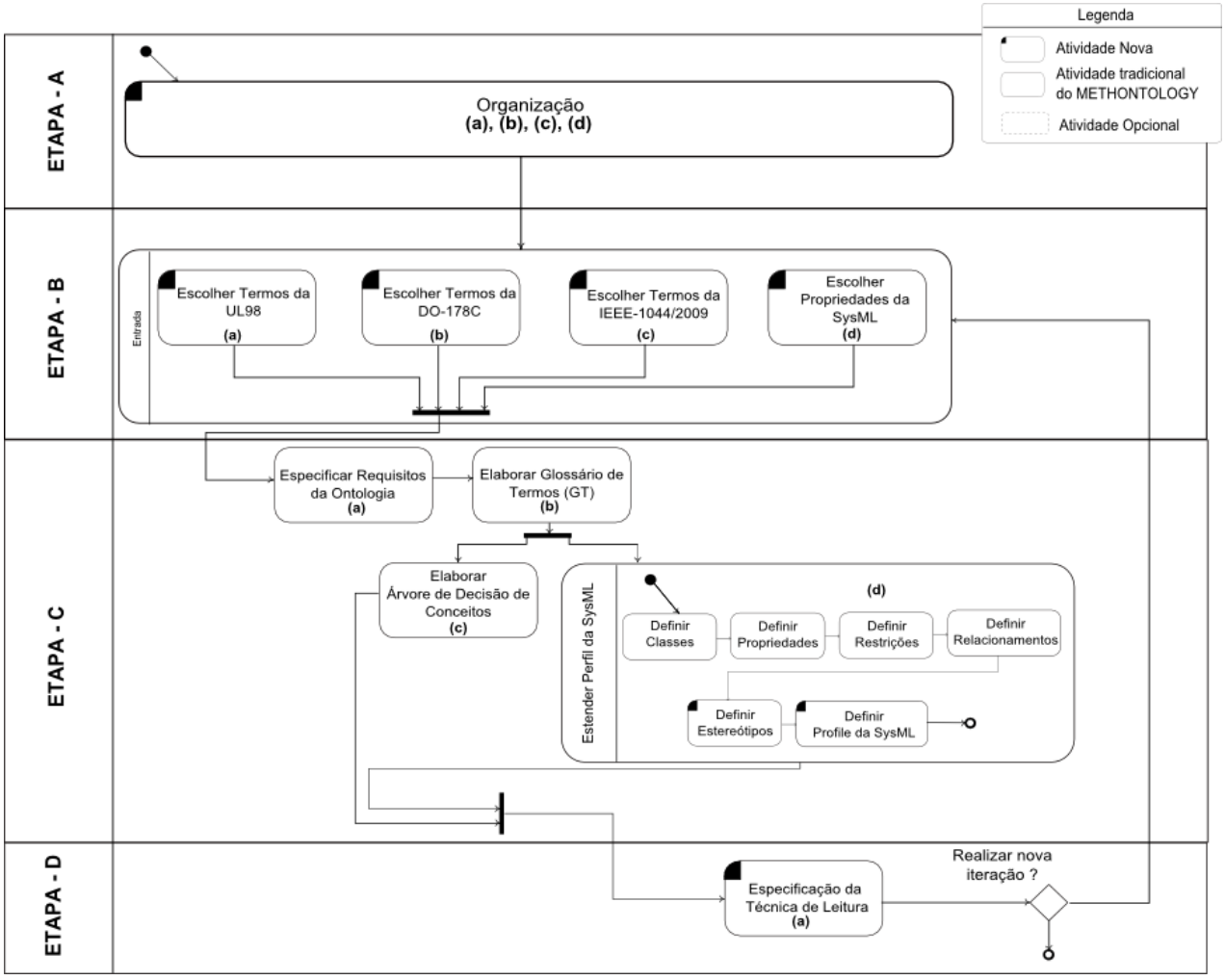


Figura 3.2 - Processo de definição das técnicas de leitura.

O processo de definição das técnicas RTSS apresentado na Figura 3.2, foi inspirado na METHONTOLOGY(FERNANDEZ-LOPEZ et al., 1997), que objetiva estabelecer as Diretrizes para a elaboração de uma ontologia seguindo etapas bem definidas. Assim, no caso da definição das técnicas de leitura, o processo que compõe a METHONTOLOGY foi utilizado com base em:

(i) para Especificar Requisitos da Ontologia, uma vez que tal atividade serve pra especificar os objetivos gerais de uma ontologia bem como as fontes de interesse para a construção dessa ontologia. Assim no contexto da definição de uma técnica de leitura ela tal atividade foi usada de modo análogo para dar suporte à especificação dos requisitos para se construir uma das cinco técnicas de leitura;

(ii) para Elaborar Glossário de Termos (GT), que no caso da definição de uma ontologia serve para definir os termos de interesse da ontologia na forma de um glossário. Assim, no contexto da definição de uma técnica de leitura tal atividade tem o objetivo de definir um glossário de termos descrevendo estereótipos e as restrições de uso na linguagem SysML;

(iii) para elaborar Árvore de decisão de conceitos, que no caso da definição de uma ontologia serve para construir uma árvore de decisão derivada a partir dos termos definidos em um glossário de termos, no contexto da definição de uma técnica de leitura tal atividade tem o objetivo de elaborar uma árvore de decisão baseada nos termos, estereótipos e restrições de uso da estrutura da linguagem SysML (i.e., aspectos sintáticos e semânticos da linguagem); e

(iv) para elaborar Tabela de Dados em que para a definição de uma ontologia tal atividade é utilizada para especificar classes, atributos e propriedades. Assim, no contexto da definição das técnicas de leitura, tal atividade tem o objetivo de definir rigorosamente os estereótipos utilizados bem como gradativamente estender o perfil da linguagem SysML para um novo conjunto de estereótipos.

Como mencionado, o processo de construção das técnicas de leituras envolve o METHONTOLOGY e incorpora outras atividades e etapas pertinentes a elaboração das técnicas de leitura. Assim, na Figura 3.2, foi apresentada uma visão *top-down* do processo estabelecido para a definição das técnicas RTSS e com auxílio de uma marca em preto no canto superior direito estão sendo representadas também as novas atividades estabelecidas para a definição de técnicas RTSS. Por exemplo, a atividade Definir estereótipos na Etapa C é adicional ao METHONTOLOGY.

A seguir serão apresentadas as etapas e atividades envolvidas na definição das técnicas de leitura a partir do processo definido.

- **Etapa A:**

A Etapa A é uma etapa preliminar para a elaboração das técnicas de leitura. Nesta etapa as fontes de defeitos foram organizadas previamente para então serem consideradas na elaboração das técnicas RTSS. Ressalta-se que para cada uma dessas fontes chegou-se a um conjunto de estereótipos que são utilizados para anotar os tipos de defeitos encontrados pelas técnicas de leitura propostas.

a) UL98: o intuito em usar essa norma de certificação foi antecipar defeitos existentes no código, tentando detectá-los em diagramas que retratam a transcrição de informações advindas diretamente de um documento de requisitos. A UL-98 é uma norma ampla e de propósito geral, pois pode ser aplicada para diferentes tipos de SEs. Para fazer essa análise antecipada dos defeitos, foi realizado um mapeamento dos defeitos tratados pela norma, o qual está apresentado na Tabela 3.1

Tabela 3.1 - Mapeamento da UL-98.

Conceito	Cobertura exigida pela Norma UL1998 (†)	Módulo do Sistema Embarcado Analisado (†)	Palavras-chave	Estereótipo
Registrador	CPU Register	CPU Registers	Temperature Light Switch Counters	«UL98CPURegister»
	Program Counter	CPU Program Counter	Start	«UL98CPUProgramCounter»
Interrupção	Interrupt	Interrupt handling and execution	Frequency PWM Serial Communication	«UL98Interrupt»
	TimeSlot Monitoring	Clock	Time Clock	«UL98Clock»
Memória	Non volatile memory	Non volatile memory	External Memory Static data	«UL98ExternalMemory»
	Periodic static memory	Volatile Memory	Volatile data it	«UL98VolatileMemory»
	Write protection (EEPROM) and protocol	Memory address External Communication Data External Communication Addressing	External Memory	«UL98ProtectMemory»
Entrada Saída	Plausibility	Input/output periphery digital Input/output Periphery analogical Input/output periphery analogical multiplexer	Switch Counters Temperature Light Signal Signal Variable Sensors Actuators Motor Module Accelerometer	«UL98PlausibilitCheck»

(†) Campos utilizados no FMEA

Para estruturar a Tabela 3.1, foi utilizado o método de análise de defeitos e falhas para SEs *Failure Method and Effect Analisys* (FMEA) (PENTTI; ATTE, 2002). O FMEA é um método amplamente utilizado pela comunidade de SEs e frequentemente empregado como a primeira etapa na análise de confiabilidade de SEs. No método FMEA é possível organizar e registrar componentes, módulos e subsistemas que serão submetidos a atividades de garantia de qualidade. No contexto deste trabalho, o método FMEA foi utilizado como inspiração para a definição das colunas da Tabela 3.1. Assim, além das colunas "Cobertura Exigida pela Norma UL-98" e "Módulo do SE", já existentes no FMEA, para efeito de retratar o mapeamento, foram adicionadas as colunas "Conceito", "Palavras-chave" e "Estereótipo".

O mapeamento foi construído da seguinte forma: a partir da cobertura exigida pela norma e do módulo do sistema em que a cobertura deveria ser avaliada, definiram-se as palavras-chave que caracterizam a cobertura de tais módulos. As palavras-chave foram extraídas a partir de um modelo UML/MEF que expressava aspectos comportamentais genéricos de um controlador de tensão variável. Nesse modelo foram identificadas palavras-chave de interesse e que caracterizavam a cobertura, de tal forma que fosse possível guiar o inspetor durante a identificação de defeitos. Essas palavras-chave são palavras que correspondem a termos genéricos que são frequentemente encontrados na especificação de SEs. Uma vez definidas as palavras-chave, elas foram agrupadas em conceitos. Finalmente, a partir do nome da cobertura exigida pela norma

UL-98, foram gerados os rótulos preliminares dos estereótipos a serem utilizados pelas técnicas de leitura. Como exemplo, o estereótipo «UL98CPURegister»foi gerado a partir do nome da cobertura existente na UL-98. Para promover maior clareza no uso dos estereótipos, em particular, para as técnicas que usam a fonte UL-98, foi utilizado o formato "UL98 + Módulo".

Ressalta-se que o conjunto de termos definidos na UL-98, para a identificação de defeitos, consiste em uma taxonomia inicial selecionada a partir dos domínios de aplicações dos exemplos experimentais explorados nos Capítulos 5 e 6. Contudo, embora inicial, tal taxonomia pode ser especializada para outros domínios de aplicações visando uma maior cobertura na identificação de defeitos.

b) DO-178C: o intuito em usar essa norma de certificação foi também antecipar a detecção de defeitos que são investigados no código, explorando defeitos de estruturas condicionais. Foram utilizados os documentos (DANIELS, 2011 e HAYHURST et al., 2001), visto que nesses documentos encontram-se práticas de uso dessa norma. Da mesma forma que para a UL-98, foi feito um mapeamento da seguinte maneira: identificou-se a cobertura mínima exigida pela norma, bem como as palavras-chave de interesse que descreviam tal cobertura. A partir da cobertura exigida e palavras-chave identificadas, foi gerado, neste caso, um único conceito denominado *Decisão*. Tal mapeamento (Tabela 3.2) é utilizado para explorar a identificação e antecipação de defeitos em artefatos IBD e Simulink. Tais artefatos foram considerados, visto que ambos apresentam elementos sintáticos semelhantes como, por exemplo, portas lógicas, condicionais e blocos funcionais. Os estereótipos gerados seguem o formato "DO178+Palavras-chave".

Tabela 3.2 - Mapeamento da DO-178C.

Conceito	Cobertura exigida pela Norma DO-178	Palavras-chave envolvidas na cobertura	Estereótipo
Decisão	Statement coverage	Logical input Gate Input Switch	«DO178Input» «DO178Gate»
	Decision coverage		
	Condition coverage		
	Condition/Decision Coverage		
	MC/DC		
	Multiple Condition Coverage		

Vale destacar, que as atividades (a) e (b) descritas acima levam em consideração normas de certificação em que a sobreposição de elementos para a identificação de defeitos não ocorre. Esse fato é importante, uma vez que o uso de tais elementos será amplamente utilizado pelas técnicas de leitura.

Assim como a UL-98, o conjunto de termos definidos para a DO-178C consiste em uma taxonomia selecionada com auxílio da abordagem apresentada a seguir. Vale destacar que tal abordagem pode ser utilizada para a identificação de outros termos além dos descritos neste capítulo.

c) Taxonomia IEEE STD1044:2009:o intuito em usar o documento *IEEE Standard Classification for Software Anomalies* (IEEE, 2010) foi o de reutilizar os valores de alguns atributos para construir estereótipos que pudessem caracterizar o tipo dos defeitos encontrados nos diversos diagramas tratados pelas técnicas de leitura. A Tabela

3.3 apresenta os atributos do documento IEEE STD1044:2009 juntamente com os respectivos valores que deram origem aos estereótipos definidos para classificar os defeitos encontrados com as técnicas e que são pertinentes no contexto de SEs e dos diagramas tratados neste trabalho.

Tabela 3.3 - Mapeamento da IEEE STD1044:2009.

Atributo da Taxonomia IEEE 1044:2009	Valor do Atributo na Taxonomia IEEE 1044:2009	Estereótipo
Detection Activity	Requirement Design Codding	«IEEERequirementMissing» «IEEEDesignMissing» «IEEECoddingMissing» «IEEESyntaxMissing» «IEEESyntaxExtra» «IEEESyntaxWrong»
Mode	Missing Extra Wrong	
Type	Syntax	
Priority	High Medium Low	«IEEEHighPriority» «IEEEMediumPriority» «IEEELowPriority»
Severity	Critical Major Minor Inconsequential	«IEEECriticalSeverity» «IEEEMajorSeverity» «IEEEMinorSeverity» «IEEEInconsequentialSeverity»

Dado o propósito de uso do documento IEEE, selecionaram-se os seguintes atributos: *Detection Activity*, *Mode*, *Type*, *Priority* e *Severity* pois cada um desses atributos fornece a base para a classificação de falhas e defeitos em geral. Cada um desses atributos é aplicável a qualquer tipo de software incluindo, por exemplo, firmware. Assim, a Tabela 3.3 foi elaborada utilizando a seguinte estratégia: extraíram-se do documento da IEEE os atributos relacionados e os valores pertinentes aos defeitos a serem tratados pelas técnicas de leitura. Dessa forma, a partir da identificação de valores relacionados a um mesmo atributo (e.g., *Detection Activity*) foi gerado o estereótipo correspondente a partir do atributo e valor. Os estereótipos foram produzidos utilizando-se o formato "IEEE + Valor + Atributo".

d) Propriedades da linguagem SysML: o intuito em usar as propriedades existentes na linguagem SysML deve-se aos seguintes motivos. O primeiro deles deve-se ao uso de elementos sintáticos. Dessa forma, utilizando-se como apoio, os recursos e propriedades da linguagem SysML, podem-se definir as primeiras técnicas de leitura (T3 e T4) para a detecção de defeitos de concorrência e hierarquia. O segundo motivo, diz respeito ao tipo de transcrição entre pares dos diagramas da SysML. Assim, a partir das estruturas sintáticas da SysML, a técnica de leitura detecta defeitos sintáticos e de transcrição (de um modelo para outro) evitando que tais defeitos sejam propagados para o código fonte. Vale destacar, que a detecção de defeitos de concorrência é de vital importância em se tratando de um SE, tais defeitos, quando presentes em um SE, podem impactar, por exemplo, no desempenho da aplicação, no mau funcionamento devido a acesso indevido de recursos compartilhados.

Para introduzir aspectos relacionados às características sintáticas e semânticas da linguagem SysML foram utilizados os documentos *UML Superstructure e SysML Spec* (OMG, 2010, 2011c). A partir desses documentos, foi possível estruturar a Tabela 3.4 com artefatos e propriedades da SysML relacionados à concorrência e hierarquia.

Tabela 3.4 - Propriedades extraídas da SysML.

Artefato	Propriedade	Estereótipo
Requirement Diagram (SysML)	Essential Requirement Technical Requirement Parallel Requirement	«EssReq» «TecReq» «ParallelReq» «NonParallelReq»
State Machine Diagram (SysML)	Hierarchy	«SuperState» «SubState»
Activity Diagram (SysML)	Parallel	«Parallel» «NonParallel»

Para estruturar a Tabela 3.4 foram selecionados pares dos diagramas - Diagrama de Requisitos/Máquina de Estados e Diagrama de Requisitos/Diagrama de Atividades - pertinentes a detecção de defeitos relativos à hierarquia e concorrência, respectivamente.

O diagrama máquina de estados foi escolhido pois representa a hierarquia através de super/subestados. Desse modo, a partir de tal representação sintática, a técnica de leitura avalia se a transcrição de um super/subestadoestá presente na especificação de um diagrama de requisito e vice-versa (se um requisito técnico/essencial está presente na máquina de estados.

No caso de defeitos de concorrência, o Diagrama de Atividades foi escolhido, pois na linguagem SysML, a concorrência é representada sintaticamente com auxílio da palavra reservada *par*. Desse modo, a técnica de leitura se apoia no uso de tal marcador sintático para auxiliar na detecção de defeitos de concorrência na transcrição de informação entre o diagrama de requisitos e o diagrama de atividades.

A Tabela 3.4 foi elaborada utilizando a seguinte estratégia: extraíram-se do documento da OMG os diagramas pertinentes aos defeitos a serem tratados pelas técnicas de leitura e em seguida foram selecionadas as propriedades de interesse. Assim, a partir da identificação de propriedades relacionadas a um mesmo par dos diagramas foi gerado o estereótipo correspondente.

- **Etapa B:**

Considerando-se que a técnica de leitura T4 foi utilizada como referência para a definição do processo da Figura 3.2, as técnicas de leitura T1, T2, T3 e T5 foram definidas a partir da identificação das fontes de defeitos apropriadas (i.e., UL-98, DO-178C, Concorrência da SysML e Taxonomia IEEE1044:2009).

Para realizar essa atividade, os termos, palavras-chave e estereótipos de interesse foram selecionados. A adoção de uma ou mais atividades depende do objetivo da técnica de leitura que se pretende construir.

- **Etapa C:**

As principais atividades realizadas de acordo com o processo apresentado na Figura 3.2 são:

a) Especificar requisitos da Ontologia: essa atividade teve como objetivo nortear a especificação rigorosa e padronizada da ontologia associada com a técnica de leitura em definição.

Assim, para minimizar inconsistências em decorrência do mal uso de diferentes fontes de defeitos durante a especificação das técnicas de leitura, foi elaborada uma

especificação do objetivo geral da ontologia associada com cada técnica de leitura. Um exemplo dessa especificação pode ser observado na Figura 3.3:

Documento de Especificação de Requisitos da Ontologia para a Técnica de Leitura
Domínio: Sistemas Embarcados - SEs **Conceitos definidos por:** Erik Aceiro Antonio **Propósito:** Ontologia sobre defeitos relacionados a aspectos de certificação da norma UL-98 para diagramas da linguagem SysML. Essa ontologia deve ser utilizada como apoio para a construção da Técnica T1 **Nível de formalismo:** semi-rigorosa **Escopo:** Lista de oito tipos de teste de cobertura exigidos pela norma de certificação UL-98: CPU Register Program Counter, Interrupt, TimeSlot Monitoring, Non volatile memory, Periodic static memory, Write protection (EEPROM) and protocol, Plausibility. **Fonte:** UL-98.

Figura 3.3 - *Template* para especificar os requisitos da ontologia das técnicas RTSS.

Conforme apresentado na Figura 3.3, o *template* é formado pelos seguintes campos: **Domínio**- corresponde ao domínio de aplicação das técnicas RTSS; **Conceitos definidos por** - pessoa responsável pela elaboração da especificação; **Propósito** - descrição sintética do objetivo geral de uma das técnicas da família RTSS; **Nível de formalismo** - define à forma de descrição dos requisitos, no caso das RTSS sempre semi-rigorosa; **Escopo** - define uma lista de *subpartes* de interesse relativas aos termos da UL-98, DO-178C, IEEE STD 1044:2009 e propriedades da SysML; e **Fonte** - define a fonte de utilizada para compor a técnica que pode ser UL-98, DO-178C, IEEE STD 1044:2009 e propriedades da SysML. Ressalta-se que tal especificação é feita apenas uma vez para cada técnica de leitura.

b) Elaborar Glossário de Termos (GT): nessa atividade foi elaborada uma especificação rigorosa dos termos utilizados em cada técnica de leitura. Para atingir esse objetivo, a partir das fontes de defeitos produzidas na etapa preliminar (Etapa A do processo da Figura 3.2), os estereótipos previamente elaborados foram especificados com auxílio rigorosa de lógica de predicados. Tal estratégia é útil, visto que determinados tipos de estereótipos devem ser aplicados a termos (palavras-chave) específicos ao longo dos diagramas SysML. Como exemplo, a Tabela 3.5 retrata a especificação rigorosa dos estereótipos «UL98CPURegister» e «UL98CPUProgramCounter».

Assim, em tal especificação teve-se a preocupação de formalizar em que momento (em qual artefato; e quais termos deveriam ser utilizados) da técnica tais estereótipos deveriam ser utilizados. Desse modo, observando-se a primeira linha da tabela, pode-se perceber que tal estereótipo deve ser aplicado apenas em contextos específicos, que levem em consideração o uso de termos como “Short” ou “Switch”, por exemplo.

Tabela 3.5 - Exemplo de Glossário de Termos para a especificação rigorosa.

Estereótipo	Restrição de uso em SysML
«UL98CPURegister»	∃ hasContextOf **some** (Short ∨ Switch ∨ Light ∨ Temperature) ∀ hasRegisterTypeOf **only** GeneralRegister
«UL98CPUProgramCounter»	∃ hasContextOf **some** Clock ∀ hasRegisterTypeOf **only** SpecificRegister

Observe que de acordo com o método METHONTOLOGY, a próxima etapa para definição da ontologia é a construção da árvore de decisão. Assim, é recomendado que as restrições de uso sejam especificadas com o auxílio de lógica de predicados.

c) Elaborar Árvore de Decisão de Conceitos: a árvore de decisão fornece um importante ponto para a modelagem na técnica de leitura. Uma árvore de decisão consiste em uma hierarquia de nós decisórios (também conhecidos como nós internos ou intermediários) e externos (nós folhas), ambos conectados por ramos. Em particular, a árvore de decisão fornece uma visão gráfica dos caminhos alternativos (condições e aspectos semânticos) previamente estabelecidos no passo anterior (b). Dessa forma, antes da especificação rigorosa da técnica, a elaboração da árvore de decisão fornece subsídios para que o projetista da técnica de leitura possa avaliar, por exemplo, se determinada condição é viável, ou então, se determinado estereótipo de fato deve ser incorporado ao longo da técnica. Um exemplo da árvore de decisão gerada neste trabalho é ilustrado na Figura 3.4.

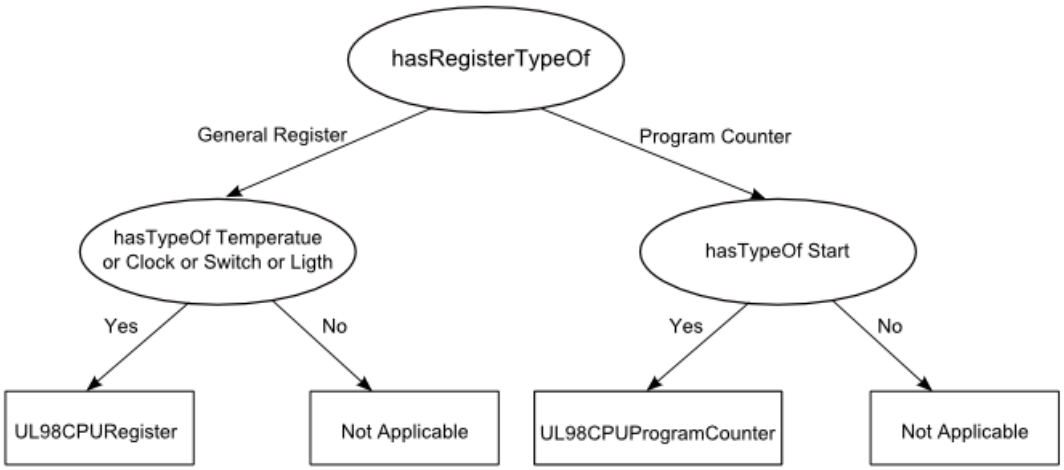


Figura 3.4 - Árvore de decisão de conceitos para a técnica T1.

A Figura 3.3 retrata uma árvore de decisão binária em que cada nó decisório dividiu-se em exatamente dois nós descendentes (nó esquerdo e direito). Na Figura 3.4 os nós decisórios estão retratados na forma de elipses e os nós folhas na forma de retângulos. Pode-se notar também, que existe um ponto de decisão "hasInterruptTypeOf" que pode ser de dois tipos "Interrupt Handler" ou "Interrupt Clock".

Em suma, o uso da árvore de decisão ao longo da definição das técnicas RTSS promove a especificação consistente das regras "SE-ENTÃO" retratadas nas técnicas. Assim, é possível antecipar e prever os diferentes caminhos condicionais que a técnica de leitura pode conter, antes de se iniciar a especificação rigorosa da técnica de leitura em questão.

d) Estender o Perfil da SysML: a realização desse passo implica na extensão do perfil da linguagem SysML uma vez que cada técnica de leitura possui um conjunto de estereótipos utilizados para se fazer a marcação dos artefatos individualmente e a comparação dos mesmos, de acordo com a estrutura das técnicas de leitura, como explicado a seguir na Etapa C. Em geral, a preparação dos artefatos e comparação dos mesmos durante a aplicação de uma técnica de leitura, envolve marcações que não

seguem um rigor sintático e semântico de uma linguagem. Tal abordagem pode levar a dificuldades, por exemplo, no processo de automação das atividades de V&V, uma vez que a carência de marcações sem um rigor sintático e/ou semântico de uma linguagem pode impactar na rastreabilidade automática (via o aporte de uma ferramenta) dos artefatos previamente inspecionados.

Para minimizar tal dificuldade, as técnicas de leitura propostas levam em consideração ao longo de sua formulação, os estereótipos previamente definidos para dar suporte à marcação dos diagramas SysML/Simulink.

Assim, para gerar um conjunto de estereótipos que pudessem ser aplicados em diagramas específicos da linguagem SysML, respeitando a sintaxe e a semântica de tal linguagem, um novo perfil foi criado a partir do perfil básico da SysML, à medida que cada técnica de leitura foi definida. Esse perfil éilustrado na Figura 3.5, através de uma visão *top-down*. Nessa visão, estão sendo retratados dois pacotes básicos da SysML ("UML4SysML" e "SysML"), tais pacotes foram incorporados em um novo pacote ("SysMLReading Technique Toolkit") via a operação de "import". Neste novo pacote gerado, estão incluídos todos os estereótipos gerados para uso ao longo da aplicação das técnicas de leitura RTSS. Na Figura 3.6 é ilustrado um recorte desse pacote, com destaque para o perfil IEEE STD 1044:2009 gerado.

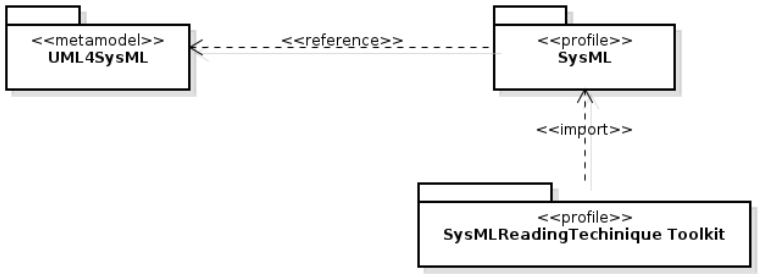


Figura 3.5 - Perfil criado em SysML para uso dos estereótipos.

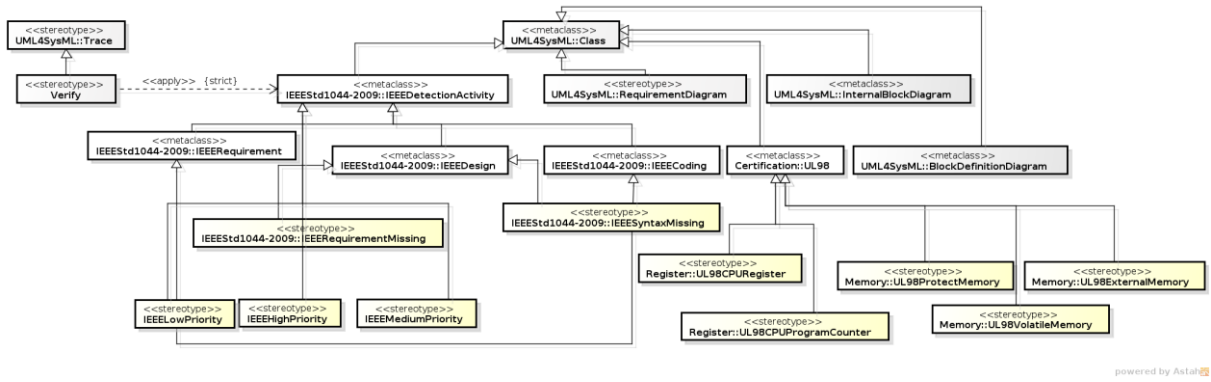


Figura 3.6 – Exemplo de Perfil para IEEE STD 1044:2009 gerado.

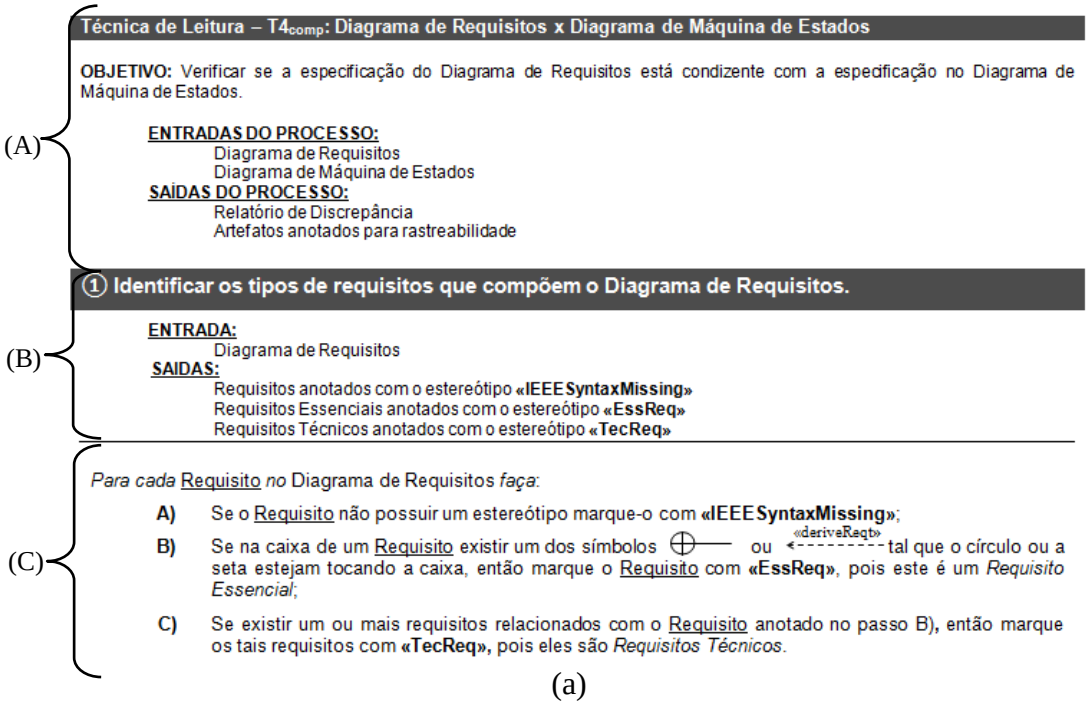
Como mencionado, a Figura 3.6 apresenta um recorte do perfil da IEEE STD 1044:2009 gerado a partir do processo definido para construção das técnicas RTSS (Figura 3.2). Na Figura 3.6 é possível observar as "metaclasses" e os estereótipos incluídos nesse pacote. Como exemplo, a "metaclasse" "StateDiagram" foi gerada a partir da "metaclasse" "UML4SysML"da SysML, o que significa que a classe gerada herda atributos/propriedades da SysML. De modo equivalente, podem-se observar os estereótipos "SubState" e "SuperState" definidos.

- **Etapa C:**

Na Etapa C, apenas a atividade de Especificação Rigorosa da Técnica de Leitura é realizada.

A especificação rigorosa das RTSS foi inspirada em algumas técnicas de leitura encontradas na literatura (BASILI; CALDIERA; et al., 1996 e DENGER; CIOLKOWSKI, 2003 e MARUCCI et al., 2002 e TRAVASSOS et al., 2000). Dessa forma, o *template* das técnicas definidas neste trabalho foi estruturado em três seções: Etapa I, Etapa II e Etapa III. A seção Etapa I e II tem o propósito de preparar os dois artefatos de entrada da técnica de leitura para a atividade de inspeção. Nessas duas seções, a partir das Diretrizes disponíveis na técnica, o inspetor anota nos diagramas SysML/Simulink os estereótipos previamente definidos. Finalmente, após a preparação dos dois diagramas de entrada, o inspetor segue com a comparação entre eles (Etapa III). Essa comparação tem o objetivo de avaliar se a transcrição da informação de um modelo para o outro foi corretamente passada.

Ressalta-se que durante este trabalho investigaram-se dois formatos diferentes para a escrita das técnicas RTSS: o formato textual (Figura 3.8 (a) e o formato de fluxograma (Figura 3.8 (b)). Como pode ser verificado, em ambos os formatos existem as partes (A), (B) e (C). A parte (A) especifica os objetivos gerais da técnica, os diagramas que serão inspecionados e as entradas e saídas; a parte (B) especifica o diagrama que será preparado para ser utilizado na comparação de consistência; e, finalmente, a parte (C) especifica os passos da técnica de leitura. A escolha de um desses formatos para elaborar as RTSS foi baseada no estudo experimental descrito em Antonio et al. (2014). Com base nos resultados desse estudo decidiu-se por adotar o formato textual.



(a)

Técnica de Leitura – $T4_{comp}$: Diagrama de Requisitos x Diagrama de Máquina de Estados

(A)

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama de Máquina de Estados.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama de Máquina de Estados

SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

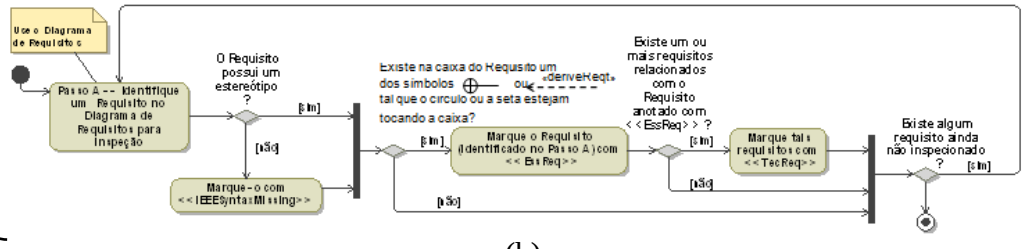
(B)

① Identificar os tipos de requisitos que compõem o Diagrama de Requisitos.

ENTRADA:
Diagrama de Requisitos

SAÍDAS:
Requisitos anotados com o estereótipo **«IEEESyntaxMissing»**
Requisitos Essenciais anotados com o estereótipo **«EssReq»**
Requisitos Técnicos anotados com o estereótipo **«TecReq»**

(C)



(b)

Figura 3.7 - *Template* usado para especificar as técnicas RTSS.

Como mencionado nas seções anteriores, às técnicas RTSS possuem três partes distintas (Etapa I, Etapa II e Etapa III). Em se tratando da identificação de defeitos, a etapa de comparação leva em consideração (além das Diretrizes inerentes da técnica) um "Formulário de Discrepâncias". Nesse formulário, são reportados os defeitos encontrados nos diagramas inspecionados. A Figura 3.8 ilustra o Formulário de Discrepâncias utilizado nas técnicas RTSS.

Formulário de Discrepância para as RTSS

Nome do Projeto		**Técnica de Leitura**	
Início da Inspeção horário em HH:MM:SS		**Data**	
Nome do Diagrama (Etapa 1)		**Nome do Diagrama (Etapa 2)**	

Tipo de Conceito		
(1) Registrador **(2)** Interrupção **(3)** Memória	**(4)** Entrada/Saída **(5)** Concorrência **(6)** Hierarquia	**(7)** Condição **(8)** Sintático **(9)** Outro

Discrepância

(1) a funcionalidade foi omitida
(2) o "projeto" é incorreto com respeito ao "diagrama de requisitos"
(3) o "projeto" é incorreto com respeito ao "diagrama Simulink"
(4) o "Simulink" é incorreto com respeito ao "diagrama IBD"
(5) o "diagrama de requisitos" é incorreto com respeito ao "projeto"
(6) outro tipo de problema encontrado (comente)

Severidade

(1) «IEEEBlockingSeverety»
as atividades envolvidas com o sistema são interrompidas até que a correção seja feita
(2) «IEEECriticalSeverity»
operações essenciais de funcionamento do sistema são inevitavelmente interrompidas, a segurança é posta em cheque e a segurança é comprometida
(3) «IEEEMajorSeverity»
operações essenciais são afetadas mas o sistema continua funcionando
(4) «IEEEMinorSeverity»
operações essenciais não são comprometidas
(5) «IEEEInconsequentialSeverity»
Não tem impacto na operação essencial do sistema

Prioridade

(1) «IEEELowPriority»
Defeitos com essa classificação serão colocados no final da fila para análise e correção
(2) «IEEEMediumPriority»
Defeitos com essa classificação serão colocados na fila de resolução após a correção dos defeitos de alta prioridade
(3) «IEEEHighPriority»
Defeitos com essa classificação serão colocados no topo da fila para análise e correção

Preencha a tabela abaixo com as informações relevantes.

Disc #	Nome do Diagrama	Tipo de Discrepância	Tipo de Conceito	Nome do Requisito	Severidade	Prioridade	Comentário
1							
2							
...		...	...		...	...	...
10							

Fim da Inspeção_____ **(horário em HH:MM:SS)**

Figura 3.8 - Formulário de discrepâncias utilizado nas técnicas RTSS.

3.3 As Técnicas RTSS e o Processo SYSMOD

Como dito anteriormente, um dos problemas no contexto de SEs é o fato dos desenvolvedores, em geral, não adotarem um processo de desenvolvimento e iniciarem o desenvolvimento da aplicação a partir de um nível (e, consequentemente, de um modelo) bem próximo ao nível de código. Em geral, na prática, esse nível é caracterizado pela elaboração dos diagramas baseados em linguagens ditas *actor-oriented* (LEE, 2004), como Simulink (MATHWORKS, 2012) e LabVIEW (ANTONIO, 2008).

Assim, com o intuito de que as técnicas RTSS pudessem ser aplicadas no âmbito de um processo de desenvolvimento, elegeu-se o processo SYSMOD (como justificado no Capítulo 1) como processo de referência para a definição das técnicas. No entanto, salienta-se que embora as técnicas tenham sido idealizadas com base nesse processo, elas podem ser usadas isoladamente, desde que o desenvolvedor tenha construído os diagramas que são tratados como entradas para essas técnicas.

Dessa forma, nesta seção apresenta-se o processo SYSMOD permeado pelas técnicas RTSS, e discute-se brevemente o que motivou a definição das técnicas conforme apresentado na Figura 3.10. Essa figura apresenta o processo SYSMOD, organizado de forma *top-down*, apresentando as técnicas de leitura RTSS (T1, T2, T3, T4 e T5) acopladas ao longo desse processo SYSMOD (ANTONIO, et al., 2014).

O objetivo dessas técnicas é estabelecer atividades de controle de qualidade e assegurar que a informação seja corretamente traduzida (transcrita) de um diagrama (modelo) para outro. Portanto, com o uso das técnicas RTSS pretende-se que os defeitos involuntariamente inseridos durante o processo de desenvolvimento possam ser identificados e corrigidos antes de serem transferidos para estágios posteriores. Assim, essa mitigação de defeitos em estágios iniciais do processo de desenvolvimento pode contribuir, por exemplo, para reduzir o retrabalho durante o desenvolvido do software embarcado.

A Figura 3.10 destaca as fases de desenvolvimento do SYSMOD que são: “Requirements”, “System Context”, “Use Cases”, “Domain Knowledge”, “System Structure” e “Dynamic System”.Cada fase do SYSMOD define o diagrama SysMLnecessário para especificação do SE. Por exemplo, na fase “Requirements”os requisitos do sistema são especificados em um Diagrama de Requisitos (REQ), e na fase “System Context”os requisitos do sistema são refinados nos diagramas *Internal Block Diagram* (IBD) e *Block Definition Diagrams* (BDD).

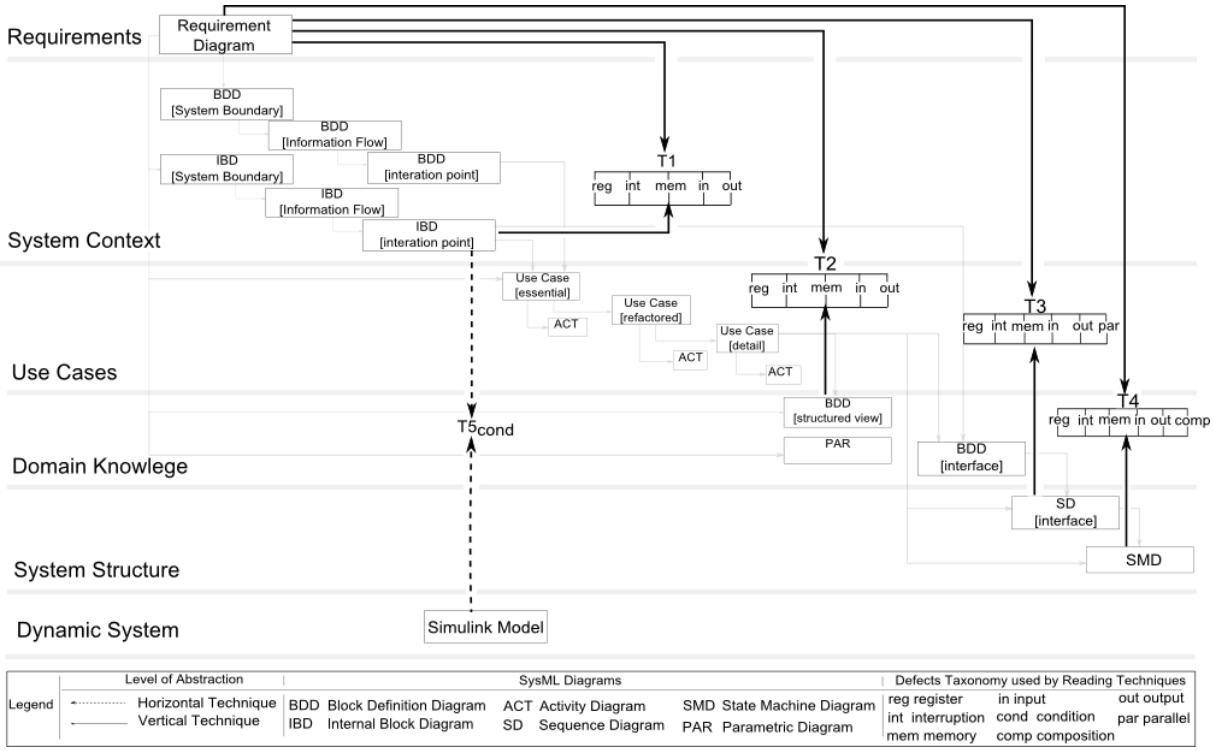


Figura 3.9 - Processo SYSMOD permeado com as técnicas RTSS.

As técnicas de leitura T1, T2, T3, T4, T5 foram estabelecidas ao longo do processo entre pares dos diagramas, nos quais a informação de um diagrama é utilizada para a construção de outro. Como exemplo, a técnica de leitura T1 é aplicada aos pares dos diagramas REQ e IBD. Além disso, a representação dessa técnica na Figura 3.10 mostra que ela possui as *subpartes* designadas por *reg, int, mem, in e out*, que correspondem às categorias da UL98.

Na Figura 3.10 as setas contínuas ligam diagramas em níveis de abstração diferentes e as setas tracejadas ligam diagramas em um mesmo nível de abstração. Dessa forma, as técnicas T1, T2, T3 e T4 utilizam diagramas em níveis de abstração diferentes (comparação vertical), enquanto que T5 utiliza o mesmo nível de abstração (comparação horizontal).

Ressalta-se que as técnicas T1, T2, T3, T4 e T5, procuram estabelecer pontos de controle (inspeção) que sejam capazes de avaliar a evolução dos requisitos do software, de acordo com a ordem natural em que os artefatos do SYSMOD devem ser elaborados. Dessa forma, após a elaboração do diagrama REQ e do diagrama IBD o inspetor pode utilizar a técnica T1 para a identificação de defeitos relativos às fases de definição de requisitos e contexto do sistema. Vale salientar que na fase de contexto do sistema ("System Context"), o projetista, ao utilizar o SYSMOD, pode efetuar pequenas re-estruturações nos diagramas IBD's. Sendo assim, considera-se que a técnica T1 deva ser usada tomando como entrada o diagrama IBD mais detalhado.

Assim, seguindo a abordagem *top-down* do SYSMOD, a próxima técnica de leitura que deve ser utilizada é a técnica T2. Essa técnica procura identificar defeitos entre a fase de definição de requisitos ("Requirements") e a fase de conhecimento de domínio ("Domain Knowlege"). Dessa forma, a técnica T2, considera o diagrama REQ e o diagrama BDD com as informações de estruturas devidamente anotadas. Ressalta-se que a etapa "Uses Cases" do processo SYSMOD é considerada opcional e, em

decorrência disso, visando minimizar os pontos de controle de qualidade associados ao SYSMOD, a técnica T2 avalia os diagramas REQ e BDD apenas.

A fase de estrutura do sistema ("System Structure"), com os diagramas de sequências (SD) e de máquinas de estados (SMD) envolve o uso das técnicas T3 e T4 respectivamente. No caso da técnica T3 são explorados tanto defeitos relativos à norma de certificação UL-98 quanto defeitos relativos a aspectos sintáticos e semânticos da linguagem SysML. Em especial, a técnica T3 visa antecipar a identificação de defeitos de concorrência e paralelismo (que por ventura possam ser introduzidos no código) para o nível de modelo. Dessa forma, utilizando-se de elementos sintáticos existentes nos diagramas SEQ da SysML, a técnica detecta tais defeitos em relação a especificação de requisitos inicial. Vale destacar, que tais defeitos são crucias para o adequado funcionamento dos SEs, uma vez que sistemas de tempo real e tempo compartilhado utilizam regiões de concorrência no código fonte.

Em se tratando da técnica T4, além dos defeitos relativos a certificação internacional UL-98, a técnica em questão procura identificar defeitos de hierarquia e composição estrutural. Tais defeitos, são mais facilmente identificados nos diagramas SMD e na especificação de requisitos inicial uma vez que a visão hierárquica de SMDs possibilita a separação de estados em super e subestados internos.

Finalmente, a técnica T5 explora defeitos existentes no modelo Simulink em relação à especificação do software, ou seja, àquilo que está especificado no diagrama REQ. Essa técnica leva em consideração a norma de certificação internacional DO-178C que visa garantir a conformidade e qualidade de sistemas embarcados. Nesse caso, a técnica T5 procura investigar pontos de decisão e estruturas condicionais especificados nos diagramas de requisitos. Tais pontos de decisão, se mal inseridos, podem impactar de forma negativa a execução dos diagramas Simulink.

Assim, com base no que foi exposto, o conjunto de técnicas de leitura RTSS procuram explorar pontos essenciais de controle, nos quais atividades de inspeção baseadas nessas técnicas podem atuar como atividades de qualidade de software, uma vez que a detecção de defeitos passa a ocorrer ao longo de todo o processo de desenvolvimento.

3.4 Considerações Finais

Neste capítulo foram apresentados dois itens de contribuição deste trabalho: (i) o processo que deu suporte à definição da família de técnicas de leitura RTSS, o qual foi extraído da própria sistemática adotada pelo autor durante a elaboração da primeira técnica definida. Esse processo promoveu a criação padronizada e organizada das demais técnicas e pode inspirar a definição de outros processos que visem outros tipos de técnicas de leitura; e (ii) o processo de desenvolvimento SYSMOD permeado pelas técnicas de leitura RTSS, gerando um processo para desenvolvimento de SEs com pontos bem definidos, nos quais atividades de inspeção apoiadas pelas técnicas RTSS podem atuar como atividades de garantia de qualidade de software.

Considerando a intenção de antecipar, para o nível de modelagem, defeitos que normalmente são detectados apenas quando já se tem o código do sistema, o processo de definição das técnicas de leitura possui uma etapa de tratamento das fontes de defeitos a serem consideradas. Tais fontes envolvem as normas de certificação UL-98 e DO-178C, uma vez que atuam no código. A UL-98 foi utilizada com o objetivo de

investigar defeitos em diagramas SysML que retratam a transcrição de informações advindas diretamente de um documento de requisitos. A norma DO-178C, também foi utilizada com o intuito de se antecipar a detecção de defeitos que exploram estruturas condicionais. Além do uso dessas normas internacionais de certificação de SEs, as técnicas RTSS exploram defeitos sintáticos e semânticos de transcrição entre pares dos diagramas da SysML relacionados a aspectos de concorrência e hierarquia. Adicionalmente a essas fontes de defeitos, foi utilizada a taxonomia de defeitos IEEE STD 1044:2009 com o intuito de reutilizar os valores de alguns atributos para organizar e construir estereótipos que pudessem caracterizar o tipo dos defeitos encontrados nos diversos diagramas tratados pelas técnicas de leitura.

Finalmente, considerando-se que desenvolvedores de SEs, em geral, iniciam o desenvolvimento a partir de um modelo bem próximo do código fonte como, por exemplo, Simulink ou LabVIEW, o processo SYSMOD foi escolhido como referência para definição e uso das técnicas RTSS. Assim, embora as técnicas de leitura tenham sido definidas com base nesse processo, elas podem ser utilizadas independentemente dele estar sendo adotado, bastando que se tenham os pares dos diagramas considerados como entradas das técnicas.

No próximo capítulo as técnicas serão apresentadas, uma a uma, seguindo-se os passos apresentados neste capítulo.

Capítulo 4
TÉCNICAS DE LEITURA RTSS

Este capítulo apresenta em detalhes as técnicas de leitura RTSS.

4.1 Considerações Iniciais

No Capítulo 3 foram apresentados os seguintes pontos: (i) o processo adotado para a definição das técnicas de leitura RTSS; e (ii) uma visão *top-down* do processo SYSMOD permeado pelas técnicas de leitura RTSS (de T1 a T5).

Assim, com base no processo estabelecido em (i) este capítulo apresenta as etapas e atividades realizadas bem como os principais artefatos gerados para a definição da família de técnicas de leitura RTSS.

Para facilitar a apresentação das seções seguintes, serão apresentadas apenas as Etapas B, C e D do processo da Figura 3.2, uma vez que a etapa preliminar ("Etapa A") de tal processo é comum a todas as técnicas. Além disso, as seções seguintes foram estruturas de maneira a facilitar a leitura, assim, em cada seção o leitor irá encontrar uma tabela fazendo referencia as etapas e atividades do processo definido em (i).

4.2 Técnica de Leitura T1 (REQ, IBD)

A técnica T1que será apresentada em detalhes explora defeitos relativos com:registradores, memória, interrupções e elementos de entrada e saída de acordo com a cobertura da norma UL-98. Tais defeitos são causados pela má formação dos diagramas REQ e IBD e, em geral, são inseridos nos diagramas involuntariamente, devido ao uso incorreto de termos (palavras-chave) ao longo do processo de desenvolvimento. Assim, tais defeitos podem ser detectados na transcrição de informação de um modelo para outro. Ressalta-se que a técnica de leitura T1 foi inserida entre as fases *Especificação de Requisitos* e *Contexto do Sistema* (vide Figura 3.9) do processo SYSMOD uma vez que nessas fases são feitas as especificações dos

pontos de interação, i.e., blocos funcionais que realizam interação com outras partes do sistema. Além disso, nesse momento são inseridas informações relativas à fronteira do sistema e fluxos lógicos.

A seguir serão apresentados os passos necessários para a construção da técnica T1 relativa à identificação de defeitos associados com a taxonomia registradores da norma UL-98.

A Tabela 4.1 apresenta as atividades realizadas em cada etapa do processo apresentado no Capítulo 3 (Figura 3.2 da Seção 3.2) para a elaboração da técnica de leitura $T1_{reg}$.

Tabela 4.1 - Etapas e atividades realizadas para elaborar a Técnica $T1_{reg}$.

Etapas	**Atividades**			
Etapa B	(a) Escolha de termos da UL-98 relativos a registradores		(c) Escolha de termos da IEEE STD 1044:2009 para classificação dos defeitos	
Etapa C	(a) Especificação dos requisitos da Ontologia	(b) Elaboração do Glossário de Termos	(c) Elaboração da Árvore de Decisão de Conceitos	(d) Extensão do Perfil da SysML
Etapa D	(a) Especificação Rigorosa da Técnica de Leitura $T1_{reg}$			

A seguir, descrevem-se, resumidamente, como as atividades mencionadas na Tabela 4.1 foram executadas para compor a técnica $T1_{reg}$:

Etapa B, atividade (a) - Escolha de termos da UL-98 relativos a registradores:

Como o intuito de usar a norma UL-98 é antecipar defeitos da norma para o nível dos diagramas, e considerando também o objetivo dessa técnica, essa atividade consistiu em consultar a Tabela 3.1 apresentada no Capítulo 3, para verificar quais palavras-chave e estereótipos estão associados com o conceito de "Registrador". As palavras-chave são: "temperature", "light", "switch", "counter" e "start", uma vez que, frequentemente, projetistas de SEs utilizam tais termos para a especificação de registradores de CPU. Consequentemente, os estereótipos associados com essas palavras-chave são: «UL98CPURegister» e «UL98CPUProgramCounter».

Etapa B, atividade (c) - Escolha de termos da IEEE1044:2009 para classificação dos defeitos:

Como o intuito de usar a norma IEEE STD 1044:2009 é estabelecer uma classificação para os defeitos identificados, essa atividade consistiu em consultar a Tabela 3.3, apresentada no Capítulo 3,com o objetivo de identificar os possíveis defeitos que poderiam ocorrer referentes a "Registradores", nos artefatos de entrada dessa técnica isoladamente (Etapa I e II), e mesmo na comparação entre esses artefatos (Etapa III). Os estereótipos que contemplam o contexto abordado por essa técnica são: «IEEERequirementMissing», «IEEEDesignMissing», «IEEEHighPriority», «IEEEMediumPriority», «IEEELowPriority» e «IEEECriticalSeverity» utilizados na técnica e no formulário de discrepância para classificação dos defeitos.O conjunto de estereótipos selecionados serve para organizar e facilitar a marcação dos defeitos como,por exemplo, o estereótipo «IEEERequirementMissing»serve para anotar um bloco de requisito no diagrama REQ que contém uma especificação ausente em relação ao projeto.

Etapa C, atividade (a) - Especificação dos requisitos da Ontologia:

Como o objetivo dessa atividade é especificar os requisitos da ontologia referente a essa técnica em particular, e considerando-se que tais requisitos são as palavras-chave e os estereótipos identificados nas duas atividades anteriores, tal especificação está apresentada na Figura 4.1. Observe que como mencionado no Capítulo 3, essa especificação possui alguns campos invariáveis, que serão iguais para todas as técnicas, e outros que são vinculados a cada técnica particularmente. Assim, no caso da técnica $T1_{reg}$ observe que o campo "Fontes de Defeitos" deve corresponder às fontes utilizadas nessa técnica que, no caso, foram a UL-98 e a IEEE STD 1044:2009. Em decorrência disso observe também que o campo "Escopo" deve fazer referência àquilo que foi usado de cada uma das fontes, para a construção da técnica. Note que da UL-98 a técnica $T1_{reg}$ explora os defeitos de cobertura "CPURegister" e "ProgramCounter", os quais correspondem aos estereótipos selecionados na Etapa B – atividade (a). O mesmo acontece com a outra fonte usada na $T1_{reg}$, isto é, a norma IEEE STD 1044:2009.

Documento de Especificação de Requisitos da Ontologia para a Técnica de Leitura$T1_{reg}$
Domínio: Sistemas Embarcados - SEs **Conceitos definidos por:** Erik Aceiro Antonio **Propósito:** Ontologia sobre defeitos relacionados ao conceito de registrador da norma UL-98 para diagramas da linguagem SysML. Essa ontologia deve ser utilizada como apoio para a construção da Técnica $T1_{reg}$ **Nível de rigor:** semi-formal **Escopo: (i)** Lista de dois tipos de teste de cobertura exigidos pela norma de certificação UL-98: "CPU Register" e "ProgramCounter"; (ii) Lista de categorias de defeitos relativos à especificação (requisitos) e ao projeto da norma IEEE STD 1044:2009. **Fonte:** UL-98; IEEE STD 1044:2009.

Figura 4.1 - *Template* de especificação dos requisitos da ontologia referente à $T1_{reg}$.

Etapa C, atividade (b) - Elaboração do Glossário de Termos:

Como o objetivo dessa atividade é a elaboração do glossário de termos, e esse glossário deve retratar os estereótipos utilizados na técnica com sua respectiva restrição de uso, então o glossário referente à técnica $T1_{reg}$ corresponde ao que está apresentado na Tabela 4.2. Por exemplo, a primeira linha da Tabela 4.2 deve ser interpretada da seguinte forma: o estereótipo «UL98CPURegister»existe apenas no contexto da aplicação das palavras-chave "short", "switch", "light" e "temperature". A outra linha da tabela pode ser interpretada de forma análoga. Similarmente a UL-98, os termos para especificação rigorosa da $T1_{reg}$, envolvendo a IEEE STD 1044:2009 também foram definidos.

Tabela 4.2 - Glossário para a especificação rigorosa para $T1_{reg}$.

Estereótipos	Restrições de uso em SysML
«UL98CPURegister»	$\exists\ hasContextOf\ \mathbf{some}\ (Short \vee Switch \vee Light \vee Temperature)$ $\forall\ hasRegisterTypeOf\ \mathbf{only}\ GeneralRegister$
«UL98CPUProgramCounter»	$\exists\ hasContextOf\mathbf{some}Start$ $\forall hasRegisterTypeOf\mathbf{only}ProgramCounter$
«IEEERequirementMissing»	$\exists\ hasContextOf\ \mathbf{only}\ InternalBlock$
«IEEEDesignMissing»	$\exists\ hasContextOf\ \mathbf{only}\ RequirementBlock$
«IEEEHighPriority»	$\forall\ hasContextOf\ \mathbf{some}\ (RequirementBlock \vee InternalBlock)$
«IEEEMediumPriority»	$\forall\ hasContextOf\ \mathbf{some}\ (RequirementBlock \vee InternalBlock)$
«IEEELowPriority»	$\forall\ hasContextOf\ \mathbf{some}\ (RequirementBlock \vee InternalBlock)$
«IEEECriticalSeverity»	$\forall\ hasContextOf\ \mathbf{some}\ (RequirementBlock \vee InternalBlock)$

Etapa C, atividade (c) - Elaboração da Árvore de Decisão de Conceitos:

Como o objetivo dessa atividade é elaborar uma árvore de decisão de conceitos, e considerando-se as restrições e condições de uso previamente definidas, a árvore de decisão que representa as possibilidades de aplicação dos estereótipos nos diagramas tratados pela técnica $T1_{reg}$ foi construída e está retratada parcialmente na Figura 4.2:

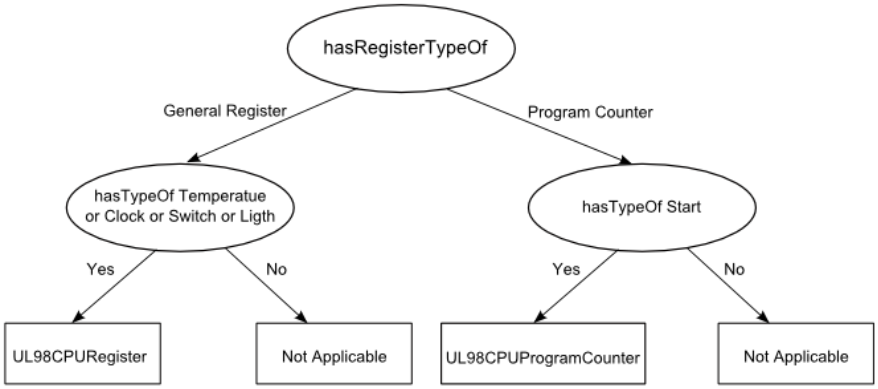


Figura 4.2 - Árvore de decisão para a técnica $T1_{reg}$.

Observe que essa árvore de decisão representa o conjunto de regras "SE-ENTÃO" que devem ser descritas na técnica de leitura $T1_{reg}$e ela auxilia na compreensão e interpretação dessas regras para a definição rigorosa da técnica.

Assim, essa árvore deve ser interpretada da seguinte forma: considerando a elipse "hasRegisterTypeOf" dois caminhos são possíveis: da esquerda quando for utilizado um registrador de propósito geral "GeneralRegister"e o da direita quando for utilizado um registrador específico "ProgramCounter". Para o caso de ser utilizado um registrador de propósito geral, outro nó decisório é avaliado, assim, se os termos forem pertinentes ao contexto da especificação do bloco dos diagramas REQ ou IBD, o bloco em questão deve ser marcado com o estereótipo «UL98CPURegister». Caso contrário, o bloco não deve receber marcação de um estereótipo.

Etapa C, atividade (d) - Extensão da SysML:

Como o objetivo dessa atividade é estender o perfil da SysML de acordo com as necessidades de cada técnica, os estereótipos e as restrições identificados nas atividades anteriores foram agrupadas e organizadas conforme a Figura 4.3.

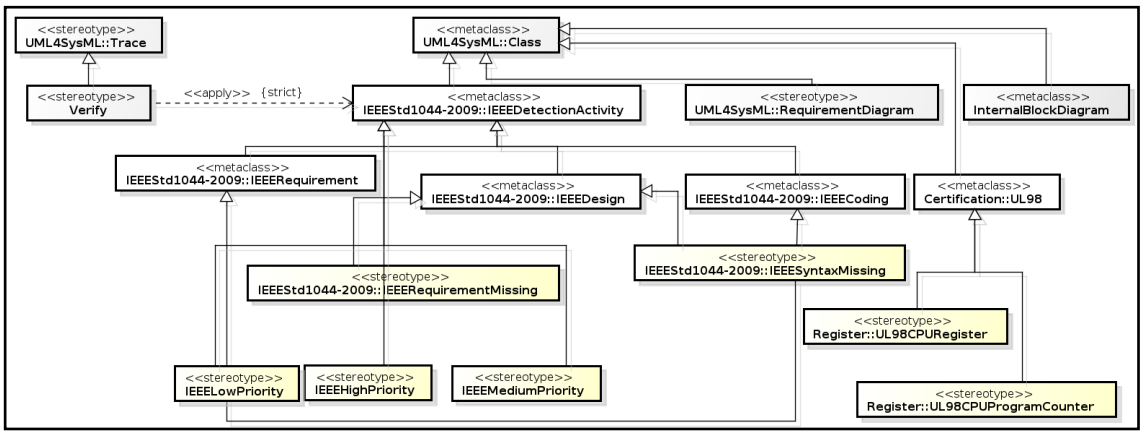


Figura 4.3 - Extensão do perfil da SysML para a técnica $T1_{reg}$.

Observe, a partir da Figura 4.3, que o estereótipo «UL98CPURegister» e «UL98CPUProgramCounter» (ambos do pacote "Register") estão associados com a metaclasse "UL98" do pacote "Certification". A partir do mecanismo de extensão indicado na Figura 4.3, através de uma linha com um triangulo na extremidade, é possível estabelecer os relacionamento entre os estereótipos definidos e os respectivos artefatos existentes da SysML. Assim, como consequência tais estereótipos podem ser aplicados as classes do metamodelo "UML4SysML" ("Requirement Diagram" e "Internal Block Diagram") da SysML. Ressalta-se que essa atividade foi realizada com o intuito de se estender o perfil da linguagem SysML para um novo conjunto de estereótipos, tendo-se como vantagem o reuso dos estereótipos para as outras técnicas de leitura, à medida que estas estejam sendo construídas.

Etapa D, atividade (a): Especificação da Técnica de Leitura:

Como o objetivo dessa atividade é a especificação da técnica de leitura, utilizando-se o *template* ilustrado na Figura 3.8 do Capítulo 3, a especificação da Técnica $T1_{reg}$ está representada nas Figuras 4.5 e 4.7.

Técnica de Leitura – $T1_{reg}$: Diagrama de Requisitos x Diagrama Interno de Blocos

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama Interno de Blocos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama Interno de Blocos
SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os requisitos do Diagrama de Requisitos com as classes para certificação de acordo com a Norma UL98

ENTRADA:
Diagrama de Requisitos
SAÍDAS:
Requisitos marcados com o estereótipo **«UL98CPUReg»**
Requisitos marcados com o estereótipo **«UL98CPUProg»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) *Se* o Requisito possuir um conceito descrevendo uma funcionalidade envolvendo um *registrador*, então faça:

A.1) Verifique se a funcionalidade associada relaciona o uso de conceitos como temperatura, luz, chaves e contadores, *então* marque o Requisito com **«UL98CPUReg»;**

A.2) Verifique se a funcionalidade associada com o requisito relaciona o uso de conceitos como inicialização, *então* marque o Requisito com **«UL98CPUProg»;**

②Identificar os blocos do Diagrama de Interno de Blocos com as classes para certificação de acordo com a Norma UL98

ENTRADAS:
Diagrama Interno de Blocos
SAÍDAS:
Blocos marcados com o estereótipo **«UL98CPUReg»**
Blocos marcados com o estereótipo **«UL98CPUProg»**

Para cada Bloco *no* Diagrama de Interno de Blocos *faça*:

B) Se o Bloco possuir um conceito descrevendo uma funcionalidade envolvendo um *registrador*, então faça:

B.1) Verifique se a funcionalidade associada relaciona o uso de conceitos como temperatura, luz, chaves e contadores, *então* marque o Bloco com **«UL98CPUReg»;**

B.2) Verifique se a funcionalidade associada relaciona o uso de conceitos como inicialização, *então* marque o Bloco com **«UL98CPUProg»;**

Figura 4.4 - Técnica $T1_{reg}$ Etapa I e II.

③ Inspecionar o Diagrama de Requisito e o Diagrama Interno de Blocos para verificar se ambos estão descrevendo a mesma funcionalidade em termos de certificação.

ENTRADAS:

Diagrama de Requisitos anotados na Etapa 1
Diagrama Interno de Blocos anotados na Etapa 2

SAÍDA:

Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os requisitos marcados com «UL98CPUReg» ou «UL98CPUProg» estão consistentes com o Diagrama Interno de Blocos.

Para cada Requisito marcado com **«UL98CPUReg»** *no* Diagrama de Requisitos *faça*:

A) Verificar se existe um Bloco marcado com **«UL98CPUReg»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Requisito marcado com **«UL98CPUProg»** *no* Diagrama de Requisitos *faça*:

B) Verificar se existe um Bloco marcado com **«UL98CPUProg»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Verificar se os blocos marcados com «UL98CPUReg» ou «UL98CPUProg» estão consistentes com o Diagrama de Requisitos

Para cada Bloco marcado com **«UL98CPUReg»** *no* Diagrama Interno de Blocos *faça*:

C) Verificar se existe um Requisito marcado com **«UL98CPUReg»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

C.1) Marque o Bloco com **«IEEEReqMissing».**

C.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Bloco marcado com **« UL98CPUProg»** *no* Diagrama Interno de Blocos *faça*:

D) Verificar se existe um Requisito marcado com **«UL98CPUProg»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

D.1) Marque o Bloco com **«IEEEReqMissing».**

D.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Figura 4.5 - Técnica $T1_{reg}$ Etapa III.

Observe que a especificação da técnica obedece o *template* apresentado na Figura 3.8 (a). Note que na parte inicial da técnica apresentam-se as três etapas da técnica, Etapa I que é a preparação do diagrama REQ, Etapa II que é a preparação do diagrama IBD e a Etapa III que é a comparação entre os diagramas previamente preparados à partir dos estereótipos definidos. Pode-se notar também, os estereótipos definidos e indicados como elementos das Diretrizes da técnica. Por exemplo, no passo A da Etapa I, pede-se para avaliar se o requisito possui um conceito descrevendo uma funcionalidade de provável representação na forma de um registrador, então caso tal registrador descreva o uso de uma das palavras-chave previamente definidas (e.g.,

temperatura e luz) o requisito em questão deve ser marcado com o estereótipo «UL98CPURegister». Vale ressaltar, que o passo A da técnica corresponde à árvore de decisão anteriormente definida (Figura 4.5 da Etapa C, atividade (c)). Nesse caso, o segundo nó decisório da esquerda na árvore é avaliado como um registrador de propósito geral. Além desse aspecto, vale comentar que as restrições de uso (Tabela 4.2 da Etapa C, atividade (b)) foram mapeadas nas Diretrizes da técnica na forma de duas condições lógicas separadas (passo A.1 e A.2) nas Diretrizes da Etapa I. As outras partes da Etapa I e II foram estruturas de maneira similar.

Em relação à Etapa III, podem-se observar as Diretrizes necessárias para realizar a comparação entre os pares de artefatos (Diagrama REQ e IBD). Nesse momento é avaliada a consistência na transformação (mapeamento) da especificação dos requisitos para o nível de modelo e vice-versa. Por exemplo, na Diretriz C é avaliado se para cada requisito previamente marcado com o estereótipo «UL98CPURegister» no REQ, existe um bloco corresponde no diagrama IBD. Assim, caso não exista uma especificação correspondente no IBD, então tal requisito deve ser marcado com o estereótipo «IEEEDesignMissing», pois tal ausência caracteriza um defeito relativo à especificação do Sistema Embarcado (SE). De modo análogo, a Diretriz E avalia se existe um bloco no modelo (IBD) que não possui correspondente no documento de requisitos, nesse caso, indicando um defeito relativo ao projeto do SE.

Em suma, essa Seção destacou os detalhes sobre a técnica de leitura $T1_{reg}$ e como ela foi construída para detectar defeitos associados com a norma de certificação UL-98 em diagramas REQ e IBD.

De modo análogo à $T1_{reg}$, as *subpartes* $T1_{int,}$ $T1_{mem,}$ $T1_{in/out}$ foram geradas e são apresentadas a seguir.

Técnica de Leitura – $T1_{int}$: Diagrama de Requisitos x Diagrama Interno de Blocos

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama Interno de Blocos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama Interno de Blocos
SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os requisitos do Diagrama de Requisitos com as classes para certificação de acordo com a Norma UL98

ENTRADA:
Diagrama de Requisitos
SAÍDAS:
Requisitos marcados com o estereótipo « **UL98Interrupt**»
Requisitos marcados com o estereótipo «**UL98Clock**»

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) Se o Requisito possuir um conceito descrevendo uma funcionalidade de *provável uso* como uma *interrupção*, então faça:

A.1) Verifique se a funcionalidade associada com a interrupção relaciona o uso de conceitos como Frequência, Multiplexação de Potência e Comunicação Serial *então* marque o Requisito com «**UL98Interrupt**»;

A.2) Verifique se a funcionalidade associada com a interrupção relaciona o uso de conceitos como tempo (*time*) ou *clock*, *então* marque o Requisito com «**UL98Clock**»;

②Identificar os blocos do Diagrama de Interno de Blocos com as classes para certificação de acordo com a Norma UL98

ENTRADAS:
Diagrama Interno de Blocos

SAÍDAS:
Blocos marcados com o estereótipo «**UL98Interrupt**»
Blocos marcados com o estereótipo «**UL98Clock**»

Para cada Bloco *no* Diagrama de Interno de Blocos *faça*:

A) Se o Bloco possuir um conceito descrevendo uma funcionalidade de *provável* uso como uma *interrupção*, então faça:

A.1) Verifique se a funcionalidade associada com a interrupção relaciona o uso de conceitos como Frequência, Multiplexação de Potência e Comunicação Serial, *então* marque o Bloco com «**UL98Interrupt**»;

A.2) Verifique se a funcionalidade associada com a interrupção relaciona o uso de conceitos como tempo (*time*) ou *clock*, *então* marque o Bloco com «**UL98Clock**»;

Figura 4.6 - Técnica $T1_{int}$ Etapa I e Etapa II.

③ Inspecionar o Diagrama de Requisito e o Diagrama Interno de Blocos para verificar se ambos estão descrevendo a mesma funcionalidade em termos de certificação.

ENTRADAS:
Diagrama de Requisitos anotados na Etapa 1
Diagrama Interno de Blocos anotados na Etapa 2

SAÍDA:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os requisitos marcados com «UL98Interrupt» ou «UL98Clock» estão consistentes com o Diagrama Interno de Blocos.

Para cada Requisito marcado com **«UL98Interrupt»** *no* Diagrama de Requisitos *faça*:

A) Verificar se existe um Bloco marcado com **«UL98Interrupt»** correspondente a este Requisito. Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Requisito marcado com **«UL98Clock»** *no* Diagrama de Requisitos *faça*:

B) Verificar se existe um Bloco marcado com **«UL98Clock»** correspondente a este Requisito. Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Verificar se os blocos marcados com «UL98Interrupt» ou «UL98Interrupt» estão consistentes com o Diagrama de Requisitos

Para cada Bloco marcado com **«UL98Interrupt»** *no* Diagrama Interno de Blocos *faça*:

C) Verificar se existe um Requisito marcado com **«UL98Interrupt»** correspondente a este Bloco. Se não existir, **isso é uma discrepância**, *então faça*:

C.1) Marque o Bloco com **«IEEEReqMissing».**

C.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Bloco marcado com **« UL98Clock»** *no* Diagrama Interno de Blocos *faça*:

D) Verificar se existe um Requisito marcado com **«UL98Clock»** correspondente a este Bloco. Se não existir, **isso é uma discrepância**, *então faça*:

D.1) Marque o Bloco com **«IEEEReqMissing».**

Figura 4.7 - Técnica $T1_{int}$ Etapa III.

Técnica de Leitura – $T1_{mem}$: Diagrama de Requisitos x Diagrama Interno de Blocos

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama Interno de Blocos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama Interno de Blocos
SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os requisitos do Diagrama de Requisitos com as classes para certificação de acordo com a Norma UL98

ENTRADA:
Diagrama de Requisitos
SAÍDAS:
Requisitos marcados com o estereótipo **« UL98ExternalMemory»**
Requisitos marcados com o estereótipo **«UL98VolatileMemory»**
Requisitos marcados com o estereótipo **«UL98ProtectMemory»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) Se o Requisito possuir um conceito descrevendo uma funcionalidade de *provável uso* como um componente de *memória*, então faça:

A.1) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Acesso à Memória Externa ou Acesso à Dados Estáticos *então* marque o Requisito com **«UL98ExternalMemory»;**

A.2) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Acesso à Dados Voláteis *então* marque o Requisito com **«UL98VolatileMemory»;**

A.3) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Memória Externa *então* marque o Requisito com **«UL98ProtectMemory»;**

②Identificar os blocos do Diagrama de Interno de Blocos com as classes para certificação de acordo com a Norma UL98

ENTRADAS:
Diagrama Interno de Blocos
SAÍDAS:
Blocos marcados com o estereótipo **« UL98ExternalMemory»**
Blocos marcados com o estereótipo **« UL98VolatileMemory»**
Blocos marcados com o estereótipo **« UL98ProtectMemory»**

Para cada Bloco *no* Diagrama de Interno de Blocos *faça*:

A) Se o Bloco possuir um conceito descrevendo uma funcionalidade *provável uso* como um componente de *memória*, então faça:

A.1) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Acesso à Memória Externa ou Acesso à Dados Estáticos *então* marque o Bloco com **«UL98ExternalMemory»;**

A.2) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Acesso à Dados Voláteis *então* marque o Bloco com **«UL98VolatileMemory»;**

A.3) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Memória Externa *então* marque o Bloco com **«UL98ProtectMemory»;**

Figura 4.8 - Técnica $T1_{mem}$ Etapa I e Etapa II.

③ Inspecionar o Diagrama de Requisito e o Diagrama Interno de Blocos para verificar se ambos estão descrevendo a mesma funcionalidade em termos de certificação.

ENTRADAS:

Diagrama de Requisitos anotados na Etapa 1
Diagrama Interno de Blocos anotados na Etapa 2

SAÍDA:

Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os requisitos marcados com «UL98ExternalMemory», «UL98VolatileMemory» ou «UL98ProtectMemory» estão consistentes com o Diagrama Interno de Blocos.

Para cada Requisito marcado com **«UL98ExternalMemory»** *no* Diagrama de Requisitos *faça*:

A) Verificar se existe um Bloco marcado com **«UL98ExternalMemory»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Requisito marcado com **«UL98VolatileMemory»** *no* Diagrama de Requisitos *faça*:

B) Verificar se existe um Bloco marcado com **«UL98VolatileMemory»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

B.1) Marque o Requisito com **«IEEEDesignMissing».**

B.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Requisito marcado com **« UL98ProtectMemory »** *no* Diagrama de Requisitos *faça*:

C) Verificar se existe um Bloco marcado com **«UL98ProtectMemory»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

C.1) Marque o Requisito com **«IEEEDesignMissing».**

C.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Verificar se os blocos marcados com «UL98ExternalMemory», «UL98VolatileMemory» ou «UL98ProtectMemory» estão consistentes com o Diagrama de Requisitos

Para cada Bloco marcado com **«UL98ExternalMemory»** *no* Diagrama Interno de Blocos *faça*:

D) Verificar se existe um Requisito marcado com **«UL98ExternalMemory»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

D.1) Marque o Bloco com **«IEEEReqMissing».**

D.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Bloco marcado com **« UL98VolatileMemory»** *no* Diagrama Interno de Blocos *faça*:

E) Verificar se existe um Requisito marcado com **«UL98VolatileMemory»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

E.1) Marque o Bloco com **«IEEEReqMissing».**

E.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

F) Verificar se existe um Requisito marcado com **«UL98ProtectMemory»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

F.1) Marque o Bloco com **«IEEEReqMissing».**

F.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Figura 4.9 - Técnica $T1_{mem}$ Etapa III.

Técnica de Leitura – $T1_{in/out}$: Diagrama de Requisitos x Diagrama Interno de Blocos

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama Interno de Blocos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama Interno de Blocos
SAIDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os requisitos do Diagrama de Requisitos com as classes para certificação de acordo com a Norma UL98

ENTRADA:
Diagrama de Requisitos
SAÍDAS:
Requisitos marcados com o estereótipo **«UL98PlausibilityCheck»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) Se o Requisito possuir um conceito descrevendo uma funcionalidade de *provável uso* como ENTRADA DE DADOS, então faça:

A.1) Verifique se a funcionalidade associada com este componente de entrada de dados descreve um elemento como, por exemplo, **Contador, Temperatura, Luz, Sinal, Sinal Contínuo, Sinal Variável, Sensor, Atuador, Motor ou Acelerômetro** *então* marque o Requisito com **«UL98PlausibilityCheck»;**

②Identificar os blocos do Diagrama Interno de Blocos com as classes para certificação de acordo com a Norma UL98

ENTRADAS:
Diagrama de Definição de Blocos

SAÍDAS:
Blocos marcados com o estereótipo **«UL98PlausibilityCheck»**
Blocos marcados com o estereótipo **«IEEEDesignWrong»**

Para cada Bloco *no* Diagrama de Interno de Blocos *faça*:

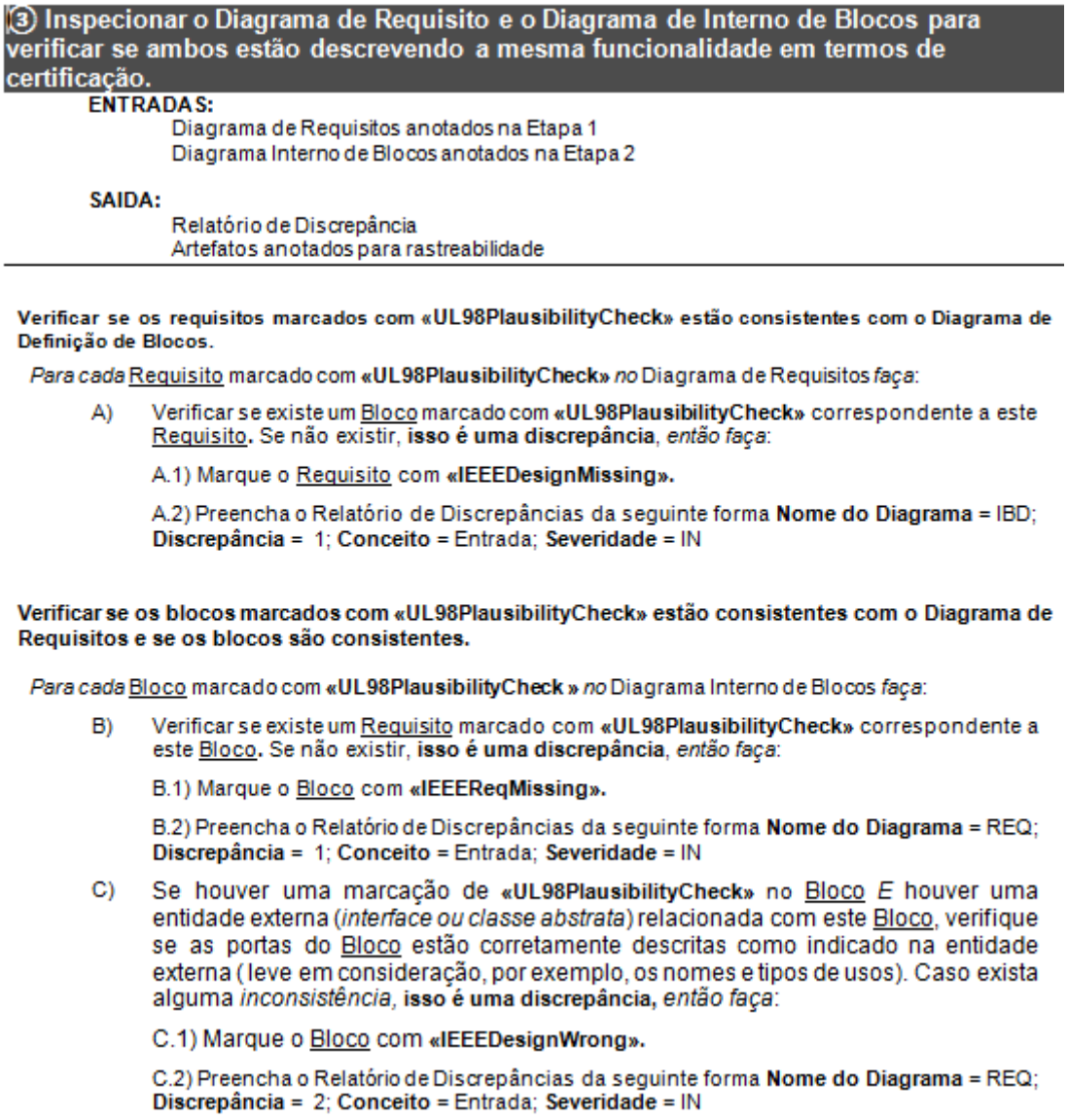
A) Se o Bloco possuir um conceito descrevendo uma funcionalidade de *provável uso* como ENTRADA DE DADOS, então faça:

A.1) Verifique se a funcionalidade associada com este componente de entrada de dados descreve um elemento como, por exemplo, **Contador, Temperatura, Luz, Sinal, Sinal Contínuo, Sinal Variável, Sensor, Atuador, Motor ou Acelerômetro** *então* marque o Bloco com **«UL98PlausibilityCheck»;**

B) Se o Bloco possuir um relacionamento com uma entidade externa (*interface ou classe abstrata*) *E* não houver marcação neste bloco de **«UL98PlausibilityCheck»** *então faça*:

B.1) Verifique se a entidade externa descreve um ou mais métodos com um *provável* uso na forma de **configuração, alteração de dados ou envio de sinal** *então* marque o Bloco com **« UL98PlausibilityCheck»;**

Figura 4.10 - Técnica $T1_{in/out}$ Etapa I e Etapa II.



③ Inspecionar o Diagrama de Requisito e o Diagrama de Interno de Blocos para verificar se ambos estão descrevendo a mesma funcionalidade em termos de certificação.

ENTRADAS:
Diagrama de Requisitos anotados na Etapa 1
Diagrama Interno de Blocos anotados na Etapa 2

SAIDA:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os requisitos marcados com «UL98PlausibilityCheck» estão consistentes com o Diagrama de Definição de Blocos.

Para cada Requisito marcado com **«UL98PlausibilityCheck»** *no* Diagrama de Requisitos *faça*:

A) Verificar se existe um Bloco marcado com **«UL98PlausibilityCheck»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Entrada; **Severidade** = IN

Verificar se os blocos marcados com «UL98PlausibilityCheck» estão consistentes com o Diagrama de Requisitos e se os blocos são consistentes.

Para cada Bloco marcado com **«UL98PlausibilityCheck »** *no* Diagrama Interno de Blocos *faça*:

B) Verificar se existe um Requisito marcado com **«UL98PlausibilityCheck»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

B.1) Marque o Bloco com **«IEEEReqMissing».**

B.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Entrada; **Severidade** = IN

C) Se houver uma marcação de **«UL98PlausibilityCheck»** no Bloco *E* houver uma entidade externa (*interface ou classe abstrata*) relacionada com este Bloco, verifique se as portas do Bloco estão corretamente descritas como indicado na entidade externa (leve em consideração, por exemplo, os nomes e tipos de usos). Caso exista alguma *inconsistência,* **isso é uma discrepância,** *então faça*:

C.1) Marque o Bloco com **«IEEEDesignWrong».**

C.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 2; **Conceito** = Entrada; **Severidade** = IN

Figura 4.11 - Técnica $T1_{in/out}$ Etapa III.

4.3 Técnica de Leitura T2(REQ, BDD)

A técnica T2explora defeitos associados com elementos de registradores, memória, interrupções e dispositivos de entrada e saída de acordo com a cobertura da norma UL-98. Tais defeitos são causados pela má formação dos diagramas REQ e BDD, e que em geral, são inseridos nos diagramas involuntariamente devidosa usos incorretos de termos ao longo do processo de desenvolvimento.

Ressalta-se que a técnica T2 foi inserida entre as fases *Requisitos* e*Conhecimento de Domínio* (vide Figura 3.9) uma vez que nesse momento da especificação do processo SYSMOD são feitas especificações nos diagramas BDD a partir do REQ. Além disso, a especificação do BDD inclui a visão geral do sistema que opcionalmente pode incluir informações relativas aos casos de uso do sistema.

Assim, tais defeitos podem ser detectados na transcrição de informação de um modelo para outro. A seguir cada uma das partes da técnica T2 é apresentada.

Técnica de Leitura – $T2_{reg}$: Diagrama de Requisitos x Diagrama de Definição de Blocos

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama de Definição de Blocos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama de Definição de Blocos
SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os requisitos do Diagrama de Requisitos com as classes para certificação de acordo com a Norma UL98

ENTRADA:
Diagrama de Requisitos
SAÍDAS:
Requisitos marcados com o estereótipo **«UL98CPUReg»**
Requisitos marcados com o estereótipo **«UL98CPUProg»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) Se o Requisito possuir um *conceito* descrevendo uma funcionalidade de *provável uso* como *registrador*, então faça:

A.1) Verifique se a funcionalidade associada relaciona o uso de conceitos como temperatura, luz, chaves e contadores, *então* marque o Requisito com **«UL98CPUReg»;**

A.2) Verifique se a funcionalidade associada com o requisito relaciona o uso de conceitos como inicialização, *então* marque o Requisito com **«UL98CPUProg»;**

②Identificar os blocos do Diagrama de Definição de Blocos com as classes para certificação de acordo com a Norma UL98

ENTRADAS:
Diagrama de Definição de Blocos
SAÍDAS:
Blocos marcados com o estereótipo **«UL98CPUReg»**
Blocos marcados com o estereótipo **«UL98CPUProg»**

Para cada Bloco *no* Diagrama de Definição de Blocos *faça*:

A) Se o Bloco possuir um *conceito* descrevendo uma funcionalidade de *provável uso* como *registrador*, então faça:

A.1) Verifique se a funcionalidade associada relaciona o uso de conceitos como temperatura, luz, chaves e contadores, *então* marque o Bloco com **«UL98CPUReg»;**

A.2) Verifique se a funcionalidade associada relaciona o uso de conceitos como inicialização, *então* marque o Bloco com **«UL98CPUProg»;**

Figura 4.12 - Técnica $T2_{reg}$ Etapa I e II.

③ Inspecionar o Diagrama de Requisito e o Diagrama de Definição de Blocos para verificar se ambos estão descrevendo a mesma funcionalidade em termos de certificação.

ENTRADAS:

Diagrama de Requisitos anotados na Etapa I
Diagrama de Definição de Blocos anotados na Etapa II

SAIDA:

Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os requisitos marcados com «UL98CPUReg» ou «UL98CPUProg» estão consistentes com o Diagrama de Definição de Blocos.

Para cada Requisito marcado com **«UL98CPUReg»** *no* Diagrama de Requisitos *faça*:

A) Verificar se existe um Bloco marcado com **«UL98CPUReg»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Requisito marcado com **«UL98CPUProg»** *no* Diagrama de Requisitos *faça*:

B) Verificar se existe um Bloco marcado com **«UL98CPUProg»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Verificar se os blocos marcados com «UL98CPUReg» ou «UL98CPUProg» estão consistentes com o Diagrama de Requisitos

Para cada Bloco marcado com **«UL98CPUReg»** *no* Diagrama de Definição de Blocos *faça*:

C) Verificar se existe um Requisito marcado com **«UL98CPUReg»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

C.1) Marque o Bloco com **«IEEEReqMissing».**

C.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Bloco marcado com **« UL98CPUProg»** *no* Diagrama de Definição de Blocos *faça*:

D) Verificar se existe um Requisito marcado com **«UL98CPUProg»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

D.1) Marque o Bloco com **«IEEEReqMissing».**

D.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Figura 4.13 - Técnica $T2_{reg}$ Etapa III.

Técnica de Leitura – $T2_{int}$: Diagrama de Requisitos x Diagrama de Definição de Blocos

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama Interno de Blocos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama de Definição de Blocos

SAIDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os requisitos do Diagrama de Requisitos com as classes para certificação de acordo com a Norma UL98

ENTRADA:
Diagrama de Requisitos

SAÍDAS:
Requisitos marcados com o estereótipo **« UL98Interrupt»**
Requisitos marcados com o estereótipo **«UL98Clock»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) Se o Requisito possuir um conceito descrevendo uma funcionalidade de *provável uso* como uma *interrupção*, então faça:

A.1) Verifique se a funcionalidade associada com a interrupção relaciona o uso de conceitos como Frequência, Multiplexação de Potência e Comunicação Serial *então* marque o Requisito com **«UL98Interrupt»;**

A.2) Verifique se a funcionalidade associada com a interrupção relaciona o uso de conceitos como tempo (*time*) ou *clock*, *então* marque o Requisito com **«UL98Clock»;**

②Identificar os blocos do Diagrama de Definição de Blocos com as classes para certificação de acordo com a Norma UL98

ENTRADAS:
Diagrama de Definição de Blocos

SAÍDAS:
Blocos marcados com o estereótipo **«UL98Interrupt»**
Blocos marcados com o estereótipo **«UL98Clock»**

Para cada Bloco *no* Diagrama de Definição de Blocos *faça*:

B) Se o Bloco possuir um conceito descrevendo uma funcionalidade de *provável uso* como uma *interrupção*, então faça:

A.1) Verifique se a funcionalidade associada com a interrupção relaciona o uso de conceitos como Frequência, Multiplexação de Potência e Comunicação Serial, *então* marque o Bloco com **«UL98Interrupt»;**

A.2) Verifique se a funcionalidade associada com a interrupção relaciona o uso de conceitos como tempo (*time*) ou *clock*, *então* marque o Bloco com **«UL98Clock»;**

Figura 4.14 - Técnica $T2_{int}$ Etapa I e II.

③ Inspecionar o Diagrama de Requisito e o Diagrama de Definição de Blocos para verificar se ambos estão descrevendo a mesma funcionalidade em termos de certificação.

ENTRADAS:
Diagrama de Requisitos anotados na Etapa I
Diagrama de Definição de Blocos na Etapa II

SAIDA:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os requisitos marcados com «UL98Interrupt» ou «UL98Clock» estão consistentes com o Diagrama de Definição de Blocos.

Para cada Requisito marcado com **«UL98Interrupt»** *no* Diagrama de Requisitos *faça*:

A) Verificar se existe um Bloco marcado com **«UL98Interrupt»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Requisito marcado com **«UL98Clock»** *no* Diagrama de Requisitos *faça*:

B) Verificar se existe um Bloco marcado com **«UL98Clock»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Verificar se os blocos marcados com «UL98Interrupt» ou «UL98Interrupt» estão consistentes com o Diagrama de Requisitos

Para cada Bloco marcado com **«UL98Interrupt»** *no* Diagrama de Definição de Blocos *faça*:

C) Verificar se existe um Requisito marcado com **«UL98Interrupt»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

C.1) Marque o Bloco com **«IEEEReqMissing».**

C.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Bloco marcado com **« UL98Clock»** *no* Diagrama de Definição de Blocos *faça*:

D) Verificar se existe um Requisito marcado com **«UL98Clock»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

D.1) Marque o Bloco com **«IEEEReqMissing».**

D.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Figura 4.15 - Técnica $T2_{int}$ Etapa III.

Técnica de Leitura – $T2_{mem}$: Diagrama de Requisitos x Diagrama de Definição de Blocos

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama de Definição de Blocos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama de Definição de Blocos
SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os requisitos do Diagrama de Requisitos com as classes para certificação de acordo com a Norma UL98

ENTRADA:
Diagrama de Requisitos
SAÍDAS:
Requisitos marcados com o estereótipo **« UL98ExternalMemory»**
Requisitos marcados com o estereótipo **«UL98VolatileMemory»**
Requisitos marcados com o estereótipo **«UL98ProtectMemory»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) Se o Requisito possuir um conceito descrevendo uma funcionalidade de *provável uso* como um componente de *memória*, então faça:

A.1) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Acesso à Memória Externa ou Acesso à Dados Estáticos *então* marque o Requisito com **«UL98ExternalMemory»;**

A.2) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Acesso à Dados Voláteis *então* marque o Requisito com **«UL98VolatileMemory»;**

A.3) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Memória Externa *então* marque o Requisito com **«UL98ProtectMemory»;**

②Identificar os blocos do Diagrama de Definição de Blocos com as classes para certificação de acordo com a Norma UL98

ENTRADAS:
Diagrama de Definição de Blocos
SAÍDAS:
Blocos marcados com o estereótipo **« UL98ExternalMemory»**
Blocos marcados com o estereótipo **« UL98VolatileMemory»**
Blocos marcados com o estereótipo **« UL98ProtectMemory»**

Para cada Bloco *no* Diagrama de Definição de Blocos *faça*:

B) Se o Bloco possuir um conceito descrevendo uma funcionalidade *provável uso* como um componente de *memória*, então faça:

B.1) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Acesso à Memória Externa ou Acesso à Dados Estáticos *então* marque o Bloco com **«UL98ExternalMemory»;**

B.2) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Acesso à Dados Voláteis *então* marque o Bloco com **«UL98VolatileMemory»;**

B.3) Verifique se a funcionalidade associada com este componente de memória relaciona o uso de conceitos como Memória Externa *então* marque o Bloco com **«UL98ProtectMemory»;**

Figura 4.16 - Técnica $T2_{mem}$ Etapa I e II.

③ Inspecionar o Diagrama de Requisito e o Diagrama Definição de Blocos para verificar se ambos estão descrevendo a mesma funcionalidade em termos de certificação.

ENTRADAS:

Diagrama de Requisitos anotados na Etapa I
Diagrama de Definição de Blocos anotados na Etapa II

SAÍDA:

Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os requisitos marcados com «UL98ExternalMemory», «UL98VolatileMemory» ou «UL98ProtectMemory» estão consistentes com o Diagrama de Definição de Blocos.

Para cada Requisito marcado com **«UL98ExternalMemory»** *no* Diagrama de Requisitos *faça*:

A) Verificar se existe um Bloco marcado com **«UL98ExternalMemory»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Requisito com **«IEEEDesignMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Requisito marcado com **«UL98VolatileMemory»** *no* Diagrama de Requisitos *faça*:

B) Verificar se existe um Bloco marcado com **«UL98VolatileMemory»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

B.1) Marque o Requisito com **«IEEEDesignMissing».**

B.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Requisito marcado com **« UL98ProtectMemory »** *no* Diagrama de Requisitos *faça*:

C) Verificar se existe um Bloco marcado com **«UL98ProtectMemory»** correspondente a este Requisito**.** Se não existir, **isso é uma discrepância**, *então faça*:

C.1) Marque o Requisito com **«IEEEDesignMissing».**

C.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Verificar se os blocos marcados com «UL98ExternalMemory», «UL98VolatileMemory» ou «UL98ProtectMemory» estão consistentes com o Diagrama de Requisitos

Para cada Bloco marcado com **«UL98ExternalMemory»** *no* Diagrama de Definição de Blocos *faça*:

D) Verificar se existe um Requisito marcado com **«UL98ExternalMemory»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

D.1) Marque o Bloco com **«IEEEReqMissing».**

D.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Para cada Bloco marcado com **« UL98VolatileMemory»** *no* Diagrama de Definição de Blocos *faça*:

E) Verificar se existe um Requisito marcado com **«UL98VolatileMemory»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

E.1) Marque o Bloco com **«IEEEReqMissing».**

E.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

F) Verificar se existe um Requisito marcado com **«UL98ProtectMemory»** correspondente a este Bloco**.** Se não existir, **isso é uma discrepância**, *então faça*:

F.1) Marque o Bloco com **«IEEEReqMissing».**

F.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Registrador; **Severidade** = IN

Figura 4.17 - Técnica $T2_{mem}$ Etapa III.

Técnica de Leitura – $T2_{in/out}$: Diagrama de Requisitos x Diagrama de Definição de Blocos

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama de Definição de Blocos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama de Definição de Blocos
SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os requisitos do Diagrama de Requisitos com as classes para certificação de acordo com a Norma UL98

ENTRADA:
Diagrama de Requisitos
SAÍDAS:
Requisitos marcados com o estereótipo **«UL98PlausibilityCheck»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) Se o Requisito possuir um conceito descrevendo uma funcionalidade de *provável uso* como ENTRADA DE DADOS, então faça:

A.1) Verifique se a funcionalidade associada com este componente de entrada de dados descreve um elemento como, por exemplo, **Contador, Temperatura, Luz, Sinal, Sinal Contínuo, Sinal Variável, Sensor, Atuador, Motor ou Acelerômetro** *então* marque o Requisito com **«UL98PlausibilityCheck»**;

②Identificar os blocos do Diagrama de Interno de Blocos com as classes para certificação de acordo com a Norma UL98

ENTRADAS:
Diagrama de Definição de Blocos

SAÍDAS:
Blocos marcados com o estereótipo **«UL98PlausibilityCheck»**
Blocos marcados com o estereótipo **«IEEEDesignWrong»**

Para cada Bloco *no* Diagrama de Definição de Blocos *faça*:

B) Se o Bloco possuir um conceito descrevendo uma funcionalidade de *provável uso* como ENTRADA DE DADOS, então faça:

A.1) Verifique se a funcionalidade associada com este componente de entrada de dados descreve um elemento como, por exemplo, **Contador, Temperatura, Luz, Sinal, Sinal Contínuo, Sinal Variável, Sensor, Atuador, Motor ou Acelerômetro** *então* marque o Bloco com **«UL98PlausibilityCheck»**;

C) Se o Bloco possuir um relacionamento com uma entidade externa (*interface ou classe abstrata*) *E* não houver marcação neste bloco de **«UL98PlausibilityCheck»** *então faça*:

C.1) Verifique se a entidade externa descreve um ou mais métodos com um *provável* uso na forma de **configuração, alteração de dados ou envio de sinal** *então* marque o Bloco com **« UL98PlausibilityCheck»**;

Figura 4.18 - Técnica $T2_{in/out}$ Etapa I e II.

③ Inspecionar o Diagrama de Requisito e o Diagrama de Definição de Blocos para verificar se ambos estão descrevendo a mesma funcionalidade em termos de certificação.

ENTRADAS:
Diagrama de Requisitos anotados na Etapa I
Diagrama Interno de Blocos anotados na Etapa II

SAIDA:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os requisitos marcados com «UL98PlausibilityCheck» estão consistentes com o Diagrama de Definição de Blocos.

Para cada Requisito marcado com **«UL98PlausibilityCheck»** *no* Diagrama de Requisitos *faça*:

D) Verificar se existe um Bloco marcado com **«UL98PlausibilityCheck»** correspondente a este Requisito. Se não existir, **isso é uma discrepância**, *então faça*:

D.1) Marque o Requisito com **«IEEEDesignMissing».**

D.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = BDD; **Discrepância** = 1; **Conceito** = Entrada; **Severidade** = IN

Verificar se os blocos marcados com «UL98PlausibilityCheck» estão consistentes com o Diagrama de Requisitos e se os blocos são consistentes.

Para cada Bloco marcado com **«UL98PlausibilityCheck »** *no* Diagrama de Definição de Blocos *faça*:

E) Verificar se existe um Requisito marcado com **«UL98PlausibilityCheck»** correspondente a este Bloco. Se não existir, **isso é uma discrepância**, *então faça*:

E.1) Marque o Bloco com **«IEEEReqMissing».**

E.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 1; **Conceito** = Entrada; **Severidade** = IN

F) Se houver uma marcação de **«UL98PlausibilityCheck»** no Bloco *E* houver uma entidade externa (*interface ou classe abstrata*) relacionada com este Bloco, verifique se as portas do Bloco estão corretamente descritas como indicado na entidade externa (leve em consideração, por exemplo, os nomes e tipos de usos). Caso exista alguma *inconsistência*, **isso é uma discrepância,** *então faça*:

F.1) Marque o Bloco com **«IEEEDesignWrong».**

F.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 2; **Conceito** = Entrada; **Severidade** = IN

Figura 4.19 - Técnica $T2_{in/out}$ Etapa III.

4.4 Técnica de Leitura T3 (REQ, SD)

A técnica $T3_{par}$ explora defeitos causados pela má formação dos diagramas REQ e SD devido ao uso incorreto de transcrições sintáticas e semânticas relativas àconcorrência.

Técnica de Leitura – $T3_{par}$: Diagrama de Sequência x Diagrama de Requisitos

OBJETIVO: Verificar se a especificação do Diagrama de Sequência está condizente com a especificação no Diagrama de Requisitos.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama de Sequencia
SAIDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar objetos que descrevem fragmentos com regiões paralelas

ENTRADA:
Diagrama de Sequencia
SAÍDAS:
Regiões marcados com o estereótipo **«NonParallel»**
Regiões marcados com o estereótipo **«Parallel»**

Para cada Fragmento *no* Diagrama de Sequencia *faça*:

A) *Se* o Fragmento (grupos de objetos que retratam um comportamento paralelo, i.e. independentes entre si) estiver rotulado com a operação **"par"**, *então* faça:

A.1) marque o fragmento em questão com o estereótipo «Parallel».

② Identificar os requisitos que descrevem semanticamente elementos paralelos

ENTRADAS:
Diagrama de Requisitos
SAÍDAS:
Blocos marcados com o estereótipo **«ParallelReq»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

B) Se o Requisito possuir um conceito descrevendo uma funcionalidade de *provável realização* em paralelo, então faça:

B.1) Verifique se a funcionalidade associada relaciona o uso de conceitos como paralelismo, concorrência, thread e recurso independente, *então* marque o Requisito com **«ParallelReq».**

Figura 4.20 - Técnica $T3_{par}$ Etapa I e II.

③ Inspecionar cada Diagrama de Sequência e Diagrama de Requisitos correspondente para verificar se os fragmentos paralelos do Diagrama de Sequência correspondem a funcionalidades descritas no Diagrama de Requisitos.

ENTRADAS:
Diagrama de Sequencia marcados na Etapa I
Diagrama de Requisitos marcados na Etapa II

SAIDA:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

Para cada Diagrama de Sequência *faça*:

C) Identificar o Diagrama de Requisitos correspondente a esse Diagrama de Sequência. Caso ele não exista, relate esse fato no Relatório de Discrepância, pois existe um Diagrama de Sequência que não está relacionado a nenhum Diagrama de Requisitos.

D) Verificar se todo requisito paralelo descrito no Diagrama de Requisitos, identificado no passo A, está representado no Diagrama de Sequência em questão:

Para cada estereótipo **«ParallelReq»** do Diagrama de Requisitos *faça*:

D.1) Se existir um bloco de objetos paralelo anotado com **«ParallelReq»** no Diagrama de Sequência, então faça um tique (√) apenas nos pares — bloco e requisito concorrente — que são *semanticamente* equivalentes.

E) Verificar se existe algum bloco anotado com **«ParallelReq»** no Diagrama de Sequência sem tique. Caso exista, então anote no bloco o estereótipo **«IEEERequirementMissing»**. Relate também esse fato no Relatório de Discrepância, pois existe um fragmento concorrente que não possui um requisito equivalente.

F) Verificar se existe algum requisito anotado com **«ParallelReq»** no Diagrama de Requisito sem tique. Caso exista, então anote no requisito o estereótipo **«IEEEDesignMissing».** Relate também esse fato no Relatório de Discrepância, pois existe um requisito que não possui um fragmento paralelo equivalente.

Figura 4.21 - Técnica $T3_{par}$ Etapa III.

Em suma, essa seção destacou os detalhes sobre a técnica de leitura $T3_{par}$ e como ela foi construída para detectar defeitos associados com a propriedade de concorrência da SysML em diagramas REQ e SD.

4.5 Técnica de Leitura T4 (REQ, MEF)

A técnica T4visaà detecção de defeitos causados pela má formação dos diagramas REQ e MEF devido ao uso incorreto de elementos de sintáticos bem como da transcrição semântica de hierarquia entre os diagramas REQ e SMD.

Técnica de Leitura – $T4_{comp}$: Diagrama de Requisitos x Diagrama de Máquina de Estados

OBJETIVO: Verificar se a especificação do Diagrama de Requisitos está condizente com a especificação no Diagrama de Máquina de Estados.

ENTRADAS DO PROCESSO:
Diagrama de Requisitos
Diagrama de Máquina de Estados
SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os tipos de requisitos que compõem o Diagrama de Requisitos.

ENTRADA:
Diagrama de Requisitos
SAÍDAS:
Requisitos anotados com o estereótipo **«IEEESyntaxMissing»**
Requisitos Essenciais anotados com o estereótipo **«EssReq»**
Requisitos Técnicos anotados com o estereótipo **«TecReq»**

Para cada Requisito *no* Diagrama de Requisitos *faça*:

A) Se o Requisito não possuir um estereótipo marque-o com **«IEEESyntaxMissing»**;

B) Se na caixa de um Requisito existir um dos símbolos ⊕— ou «deriveReqt» <---------- tal que o círculo ou a seta estejam tocando a caixa, então marque o Requisito com **«EssReq»**, pois este é um *Requisito Essencial*;

C) Se existir um ou mais requisitos relacionados com o Requisito anotado no passo B), então marque os tais requisitos com **«TecReq»**, pois eles são *Requisitos Técnicos*.

②Identificar os tipos de estados que compõem o Diagrama de Máquina de Estados.

ENTRADAS:
Diagrama de Máquina de Estados
SAÍDAS:
Estados anotados com o estereótipo **«IEEESyntaxMissing»**
Superestados anotados com o estereótipo **«SuperState»**
Subestados anotados com o estereótipo **«SubState»**

Para cada Estado *no* Diagrama de Máquina de Estados *faça*:

D) Se o Estado não possuir um rótulo (nome) marque-o com **«IEEESyntaxMissing»;**

D.1) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = MEF; **Discrepância** = 1; **Conceito**=8; **Severidade** = 3

E) Se o Estado possuir subestados, então marque-o com **«SuperState»**, pois este é um Superestado;

Para cada Estado *no* Diagrama de Máquina de Estados **NÃO MARCADO** *com* **«SuperState»** *faça*:

F) Marque o Estado com **«SubState»**, pois este é um Subestado.

Figura 4.22 - Técnica $T4_{comp}$ Etapa I e Etapa II.

③ Inspecionar o Diagrama de Requisito e o Diagrama de Máquina de Estados para verificar se ambos estão retratando a mesma composição.

ENTRADAS:
Diagrama de Requisitos anotados na Etapa 1
Diagrama de Máquina de Estados anotados na Etapa 2

SAÍDA:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os estados marcados com «SuperState» ou «SubState» estão consistentes no Diagrama de Requisitos.

Para cada Estado marcado com **«SuperState»** *no* Diagrama de Máquina de Estados *faça*:

A) Verificar se existe um Requisito marcado com **«EssReq»** correspondente a este Estado. Se não existir, **isso é uma discrepância**, *então faça*:

A.1) Marque o Estado com **«IEEERequirementMissing».**

A.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = MEF; **Discrepância** = 1; **Conceito**=HE,RE; **Severidade** = IN

Para cada Estado marcado com **«SubState»** *no* Diagrama de Máquina de Estados *faça*:

B) Verificar se existe um Requisito marcado com **«TecReq»** correspondente a este Estado. Se não existir, **isso é uma discrepância**, *então faça*:

B.1) Marque o Estado com **«IEEERequirementMissing»**

B.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = MEF; **Discrepância** = 1; **Conceito**=HE,RE; **Severidade** = IN

Verificar se os requisitos marcados com «EssReq» ou «TecReq» estão consistentes no Diagrama de Estados.

Para cada Requisito marcado com **«EssReq»** *no* Diagrama de Requisitos *faça*:

C) Verificar se existe um Estado marcado com **«SuperState»** correspondente a este Requisito. Se não existir, **isso é uma discrepância**, *então faça*:

C.1) Marque o Requisito com **«IEEEDesignMissing»**

C.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 2; **Conceito**=HE,ST; **Severidade** = IN

Para cada Requisito marcado com **«TecReq»** *no* Diagrama de Requisitos *faça*:

D) Verificar se existe um Estado marcado com **«SubState»** correspondente a este Requisito. Se não existir, **isso é uma discrepância**, *então faça*:

D.1) Marque o Requisito com **«IEEEDesignMissing»**

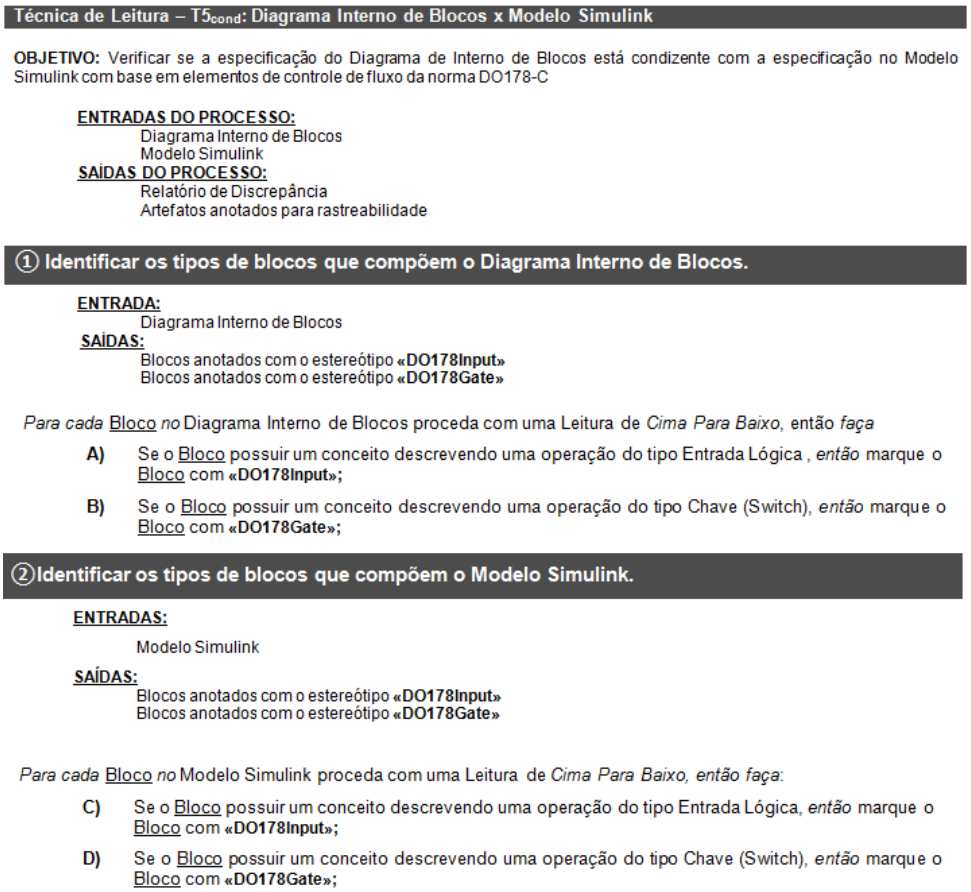
D.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = REQ; **Discrepância** = 2; **Conceito**=HE,ST; **Severidade** = IN

Figura 4.23 - Técnica $T4_{comp}$ Etapa III.

Em suma, essa Seção destacou os detalhes sobre a técnica de leitura $T4_{comp}$ e como ela foi construída para detectar defeitos associados com a composição e hierarquia em diagramas REQ e SMD.

4.6 Técnica de Leitura T5 (IBD, SML)

A técnica T5explora defeitos associados com desvios condicionais de acordo com a cobertura da norma DO-178C. Tais defeitos são causados pela inconsistência dos diagramas IBD e SML. Tal inconsistência pode ser causada, por exemplo, pela ausência de informação na descrição dos fluxos e caminhos lógicos do Simulink ou pela falta de informação adequada para a transcrição de um diagrama para outro.



Técnica de Leitura – $T5_{cond}$: Diagrama Interno de Blocos x Modelo Simulink

OBJETIVO: Verificar se a especificação do Diagrama de Interno de Blocos está condizente com a especificação no Modelo Simulink com base em elementos de controle de fluxo da norma DO178-C

ENTRADAS DO PROCESSO:
Diagrama Interno de Blocos
Modelo Simulink
SAÍDAS DO PROCESSO:
Relatório de Discrepância
Artefatos anotados para rastreabilidade

① Identificar os tipos de blocos que compõem o Diagrama Interno de Blocos.

ENTRADA:
Diagrama Interno de Blocos
SAÍDAS:
Blocos anotados com o estereótipo **«DO178Input»**
Blocos anotados com o estereótipo **«DO178Gate»**

Para cada Bloco *no* Diagrama Interno de Blocos proceda com uma Leitura de *Cima Para Baixo*, então *faça*

A) Se o Bloco possuir um conceito descrevendo uma operação do tipo Entrada Lógica , *então* marque o Bloco com **«DO178Input»;**

B) Se o Bloco possuir um conceito descrevendo uma operação do tipo Chave (Switch), *então* marque o Bloco com **«DO178Gate»;**

②Identificar os tipos de blocos que compõem o Modelo Simulink.

ENTRADAS:
Modelo Simulink
SAÍDAS:
Blocos anotados com o estereótipo **«DO178Input»**
Blocos anotados com o estereótipo **«DO178Gate»**

Para cada Bloco *no* Modelo Simulink proceda com uma Leitura de *Cima Para Baixo, então faça*:

C) Se o Bloco possuir um conceito descrevendo uma operação do tipo Entrada Lógica, *então* marque o Bloco com **«DO178Input»;**

D) Se o Bloco possuir um conceito descrevendo uma operação do tipo Chave (Switch), *então* marque o Bloco com **«DO178Gate»;**

Figura 4.24 - Técnica $T5_{cond}$ Etapa I e II.

③ Inspecionar o Diagrama IBD e o Simulink são consistentes conforme a DO-178C

ENTRADAS:

Diagrama Interno de Blocos anotados
Modelo Simulink anotados

SAÍDA:

Relatório de Discrepância
Artefatos anotados para rastreabilidade

Verificar se os blocos marcados com «DO178Input» ou «DO178Gate» estão consistentes no Modelo Simulink.

Para cada Bloco marcado com **«DO178Input»** *no* Diagrama Interno de Blocos *faça*:

E) Verificar se existe um Bloco especificado marcado no Modelo Simulink com **«DO178Input».** Se não existir, **isso é uma discrepância**, *então faça*:

E.1) Marque o Bloco no Diagrama Interno de Blocos com **«IEEECodingMissing».**

E.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = SML; **Discrepância** = 5; **Conceito**=DE; **Severidade** = 2; **Prioridade** = 1

Para cada Bloco marcado com **«DO178Gate»** *no* Diagrama Interno de Blocos *faça*:

F) Verificar se existe um Bloco especificado marcado no Modelo Simulink com **«DO178Gate».** Se não existir, **isso é uma discrepância**, *então faça*:

F.1) Marque o Bloco no Diagrama Interno de Blocos com **«IEEECodingMissing».**

F.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = SML; **Discrepância** = 5; **Conceito**=DE; **Severidade** = 2; **Prioridade** = 1

Verificar se blocos marcados com «DO178Input» ou «DO178Gate» estão consistentes no Diagrama Interno de Blocos.

Para cada Bloco marcado com **«DO178Input»** *no* Modelo Simulink *faça*:

G) Verificar se existe um Bloco especificado marcado no Diagrama Interno de Blocos com **«DO178Input».** Se não existir, **isso é uma discrepância**, *então faça*:

G.1) Marque o Bloco no Modelo Simulink com **«IEEEDesignMissing».**

G.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 3; **Conceito**=DE; **Severidade** = 2; **Prioridade** = 1

Para cada Bloco marcado com **«DO178Gate»** *no* Modelo Simulink *faça*:

H) Verificar se existe um Bloco especificado marcado no Diagrama Interno de Blocos com **«DO178Gate».** Se não existir, **isso é uma discrepância**, *então faça*:

H.1) Marque o Bloco no Modelo Simulink com **«IEEEDesignMissing».**

H.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 3; **Conceito**=DE; **Severidade** = 2; **Prioridade** = 1

I) *Caso* o Bloco exista no Diagrama Interno de Blocos *faça*:

I.1) *Se* a última porta de entrada tiver um rótulo "*", *então faça*:

I.1.1) Verifique se o número de entradas do Bloco no Simulink é mesmo do Bloco do IBD;

I.1.2) Verifique se a primeira entrada do Bloco no Simulink é consistente com a primeira entrada do Bloco do IBD, i.e, verifique se os tipos das entradas são as mesmas; e

I.1.3) Verifique se o fluxo alternativo do Bloco no Simulink é consistente com os fluxo alternativo no Bloco do IBD, i.e., verifique se o valor de indexação para vetores é especificado no IBD ou se a operação padrão (*default*) está especificada de tal forma que se possa representa-la no Simulink.

I.1.4) *Se* um dos passos anteriores (I.1.1, I.1.2 ou I.1.3) não foi atendido, *então faça*:

I.1.4.1) Marque o Bloco no Modelo Simulink com **«IEEEDesignMissing».**

I.1.4.2) Preencha o Relatório de Discrepâncias da seguinte forma **Nome do Diagrama** = IBD; **Discrepância** = 3; **Conceito**=7; **Severidade** = 3 ; **Prioridade** = 2

Figura 4.25 - Técnica $T5_{cond}$ Etapa III.

4.7 Considerações Finais

Em suma, esse capítulo apresentou a família de técnicas de leitura RTSS elaboradas a partir do processo estabelecido no Capítulo 3 (Seção 3.2). Desse modo, pode-se observar (passo a passo) os detalhes acerca da concepção de como cada uma das técnicas definidas (T1 a T5) foram concebidas. Assim, pode-se verificar que as técnicas T1 e T2 exploram defeitos relativos a aspectos de certificação, em especial, com referencia à norma de certificação internacional UL-98. Também, pode-se observar que as técnicas T3 e T4 exploram defeitos associados com aspectos pertinentes à estrutura da linguagem SysML, como por exemplo, concorrência e hierarquia. Finalmente foram apresentados os detalhes sobre as Diretrizes da técnica T5 que envolve, sobretudo, a identificação de defeitos relativos a aspectos de certificação para veículos aéreos, e para isso leva em consideração a norma internacional de certificação DO178C. Ressalta-se que esse capítulo, procurou ressaltas as subpartes $T1_{reg}$, $T2_{mem}$,$T3_{par}$,$T4_{comp}$e $T5_{cond}$, uma vez que a forma de geração das outras técnicas segue o mesmo princípio do exposto aqui.

Finalmente, no Capítulo 5 e Capítulo 6 serão apresentadas as avaliações experimentais realizadas para avaliar a viabilidade de aplicação de uma das técnicas de leitura da família RTSS bem como, a efetividade de uso de tais técnicas ao longo do processo de desenvolvimento SYSMOD.

Capítulo 5
ESTUDOS EXPERIMENTAIS

A avaliação de trabalhos da área de Engenharia de Software é uma atividade essencial, em particular, para se identificar possíveis avanços alcançados com a proposição de novas técnicas. Este capítulo apresenta os estudos experimentais conduzidos para avaliação das técnicas RTSS.

5.1 Considerações Iniciais

Com base na proposição das técnicas de leitura RTSS – definidas respectivamente nos Capítulos 3 e 4 – e considerando-se os seguintes pontos: (i) importância da avaliação de trabalhos da área de Engenharia de Software (BASILI, et al., 1994 e PRESSMAN, 2009) e (ii) necessidade de estudos de viabilidade para averiguar se um novo processo preenche os requisitos necessários para o qual ele foi criado (SHULL et al., 2002), este capítulo apresenta dois estudos de viabilidade realizados para avaliação da técnica de leitura $T4_{comp}$.

O Estudo de Viabilidade I (Seção 5.2) é uma avaliação preliminar sobre a consistência da escrita da Técnica de Leitura $T4_{comp}$, que foi a primeira técnica elaborada, como mencionado anteriormente. Neste estudo, foram coletadas evidências para formulação das demais técnicas RTSS. Dentre as informações qualitativas extraídas podem-se citar, por exemplo, as dificuldades durante a aplicação, em particular, sobre os passos da técnica e o uso dos estereótipos propostos. Além disso, neste estudo explorou-se também o formato de escrita da técnica, o qual poderia ser texto ou fluxograma. Assim, a partir desse estudo foi possível, gradativamente, avaliar e reestruturar alguns pontos de fundamental relevância sobre a técnica $T4_{comp}$, que culminaram em Diretrizes para a elaboração das outras técnicas, inclusive no que diz respeito ao processo apresentado no Capítulo 3. Desse modo, o Estudo de Viabilidade II (Seção 5.3) foi executado tendo em vista essa melhoria gradativa. Nesse estudo a técnica de leitura $T4_{comp}$ foi avaliada com respeito à *efetividade* e *eficiência* na detecção de defeitos em diagramas SysML.

A partir desses dois estudos elaborados, pontos importantes para a definição das outras técnicas RTSS foram estabelecidos. Em particular destacam-se: (i) formato texto para a elaboração das técnicas RTSS; (ii) estrutura em Seções e Partes; (iii) viabilidade de uso dos estereótipos propostos para a marcação de pares dos diagramas SysML ao longo da aplicação da técnica; e (iv) processo sistemático para dirigir a definição das demais técnicas.

Finalmente, para complementar à avaliação da técnica $T4_{comp}$, um estudo experimental para caracterizar a compreensão dos diagramas Simulink foi realizado. Esse estudo é brevemente comentado na Seção 5.4.

5.2 Estudo de Viabilidade I

A questão principal investigada nesse estudo de viabilidade foi se a técnica estava formulada de maneira consistente, isto é, se ela estava redigida de forma clara e objetiva, se os estereótipos usados para fazer as marcações estavam adequados e caracterizavam corretamente os defeitos encontrados nos artefatos avaliados bem como a *efetividade* no uso dos estereótipos. Adicionalmente a esses pontos mais relevantes, aproveitou-se esse estudo para avaliar dois possíveis formatos de escrita das técnicas RTSS – formatos texto e de fluxograma. A avaliação foi feita por meio de uma escala de pontos atribuídos pelos participantes (entre 1 e 5) à cada questão do formulário de avaliação da técnica. Foram utilizados dois diagramas distintos em SysML (OMG, 2010): um destilador de água (*AppDistill*); e um Sistema de Veículo Utilitário Híbrido (HSUV) (*AppHSUV*). A descrição desse estudo de viabilidade apresenta as principais etapas sugeridas por Wohlin (WOHLIN et al, 2000) para a condução de estudos experimentais.

a) Definição do Estudo de Viabilidade: com base no modelo *Goal-Question-Metric* (GQM) (BASILI, et al., 1994) o objetivo deste estudo de viabilidade foi definido da seguinte forma:

> **Analisar**a Técnica de Leitura $T4_{comp}$
> **Com o propósito de** avaliação da viabilidade de aplicação
> **Com respeito à** consistência na sua formulação
> **Do ponto de vista**do inspetor
> **No contexto de** alunos de pós-graduação

b) Definição das Hipóteses: para investigar a consistência da técnica, as seguintes hipóteses, nula e de pesquisa, foram formuladas:

Hipótese I:

- **H_0**: A técnica $T4_{comp}$ não está formulada de maneira consistente, i.e., a média de pontos atribuídos pelos participantes é inferior a 50%.
- **H_1**: A técnica $T4_{comp}$ está formulada de maneira consistente, i.e., a média de pontos atribuídos pelos participantes é superior ou igual a 50%.

c) Seleção do Contexto: de acordo com Wohlin et al. (2000), a seleção do contexto pode ser caracterizada por meio de quatro dimensões que são: (1) *off-line* vs. *on-line* – neste caso é *off-line*, pois o estudo não é aplicado no contexto da indústria; estudante vs. profissional – neste caso os participantes são estudantes; (3) *toy* vs. real – neste caso é um exemplo *toy*; e (4) específico vs. geral – neste caso é específico e os resultados não podem ser generalizados para outros contextos.

d) Seleção das variáveis: as seguintes variáveis independentes e variáveis dependentes foram consideradas nesse estudo:

- **Variável independente:** a técnica de leitura $T4_{comp}$
- **Variável dependente:** a média de pontos (μ), i.e., o número total de pontos atribuídos a todas as questões (entre 1 e 5) dividido pelo total de questões.

e) Seleção dos participantes: os participantes foram selecionados de acordo com a conveniência e eram alunos de pós-graduação em Engenharia de Software. Eles tinham experiência em modelagem com UML e atividades de Verificação e Validação (V&V).

f) Instrumentação: os instrumentos utilizados nesse estudo foram:

- Formulário de consentimento;
- Formulário de caracterização;
- Material de treinamento sobre a técnica $T4_{comp;}$
- Diagramas de Requisitos (REQ) e de Máquina de Estados (MEF) para serem inspecionados pela técnica $T4_{comp;}$
- Técnica de leitura $T4_{comp}$ (nos formatos texto e fluxograma);
- Formulário de defeitos; e
- Formulário de viabilidade de aplicação da técnica, composto de questões abertas e fechadas.

g) Preparação: a preparação levou em consideração a organização dos participantes e a aplicação da atividade conforme a Tabela 5.1:

Tabela 5.1 - Preparação do Estudo de Viabilidade I.

Atividades	**Descrição**	
Apresentação	Apresentações de 30 mim sobre a técnica e instruções de como os participantes deveriam responder os formulários	
Divisão dos grupos	Divisão dos grupos: G1 (13 participantes) e G2 (14 participantes	
Execução do experimento	G1	G2
	$T4_{comp}$ (Texto) AppDistill	$T4_{comp}$ (Fluxo) AppHSUV
	$T4_{comp}$ (Fluxo) AppHSUV	$T4_{comp}$ (Texto) AppDistill

f) Análise e Interpretação: Após a aplicação da técnica de leitura, os participantes preencheram o "Formulário de viabilidade" e pontuaram as questões com base na interpretação individual sobre a aplicação da técnica de leitura $T4_{comp}$. O formulário foi composto de questões abertase fechadas estabelecidas com auxílio da escala de Likert (NAVIDI, 2010). Os resultados referentes às questões fechadas estão apresentados nas Figuras 5.1 e 5.2, discriminados de acordo com os formatos utilizados (texto e fluxograma). Na Figura 5.1 apresentam-se sete questões associadas com o entendimento geral dos participantes em relação à aplicação da técnica de leitura $T4_{comp}$ e na Figura 5.2, apresentam-se dez questões referentes às propriedades utilizadas na formulação da técnica de leitura.

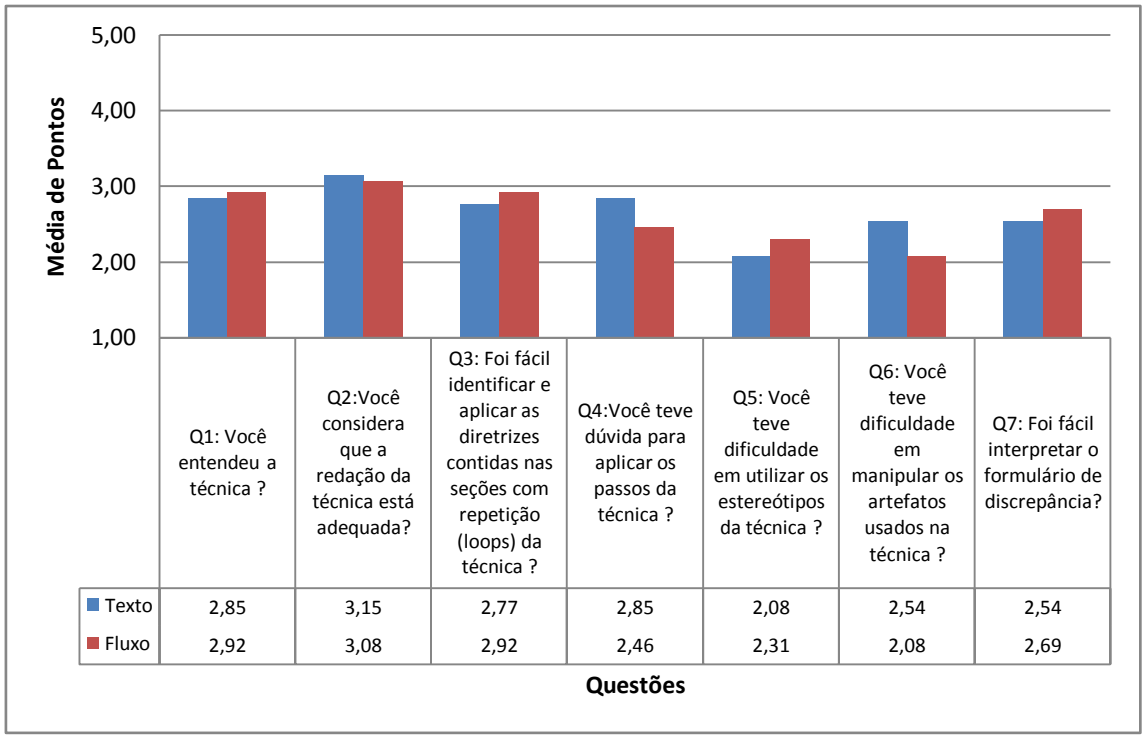


Figura 5.1 - Resultados sobre o entendimento da técnica pelos participantes.

O gráfico da Figura 5.1 ilustra a relação entre um conjunto de sete questões (Q1-Q7) e a escala de opinião individual de cada participante. Esse gráfico retrata a média de valores pontuados por cada participante quanto à consistência da técnica de leitura. Por exemplo, a questão Q1 avalia se o participante entendeu a técnica de leitura. Nesse caso obteve-se a média $\mu = 2,85$ para o formato texto e $\mu = 2,92$ para o formato fluxograma.

De modo análogo, na Figura 5.2 estão sendo apresentadas dez questões (Q1-Q10) sobre propriedades desejadas na técnica de leitura $T4_{comp}$em relação à escala de opinião. Assim, tomando-se como exemplo a questão Q5, que avalia a propriedade "Fácil entendimento", verifica-se que para essa propriedade obteve-se média $\mu = 3,38$ no formato texto e $\mu = 3,00$ no formato fluxograma.

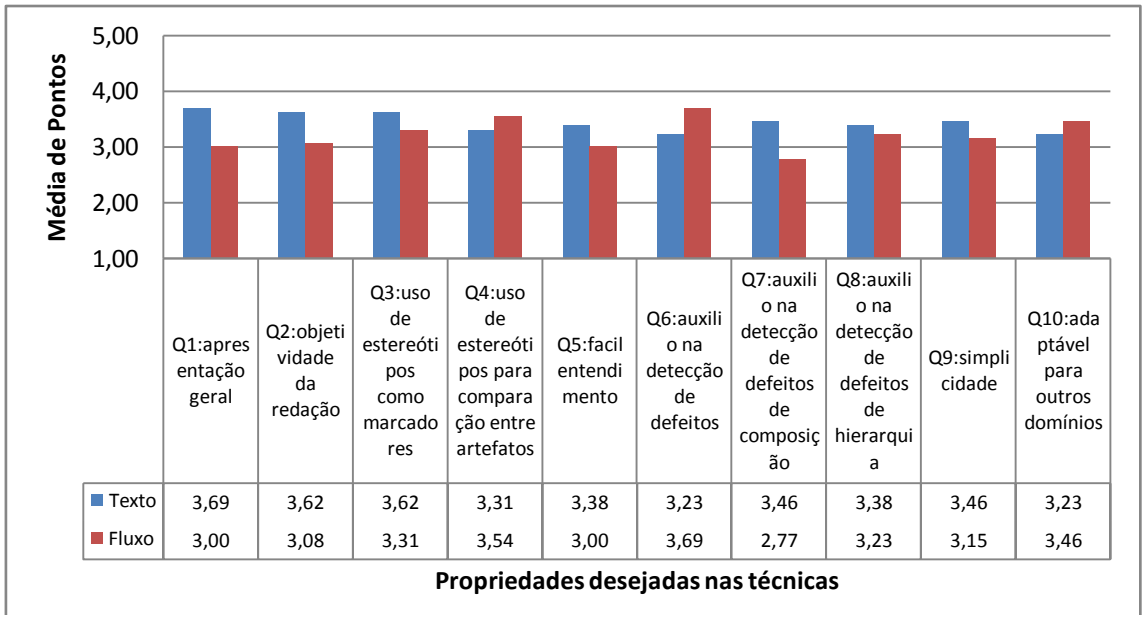


Figura 5.2 - Resultados sobre as propriedades desejadas da técnica.

Finalmente, para fornecer uma síntese geral da avaliação realizada, os dados apresentados anteriormente (Figura 5.1 e Figura 5.2) foram organizados na Tabela 5.2.

Tabela 5.2 - Resultado geral do Estudo de Viabilidade I.

	Técnica $T4_{comp}$ Formato Texto		**Técnica $T4_{comp}$ Formato Fluxo**	
	Média de pontos obtidos (A)	**Porcentagem** C=(A/5)*100	**Média de pontos obtidos** (B)	**Porcentagem** D=(B/5)*100
Média (μ)	**3,04**	**61%**	**2,94**	**59%**

A partir da Tabela 5.2, pode-se observar que a média obtida para o formato texto foi de 3,04 (61%) e para o formato fluxo foi de 2,94 (59%). A partir desses resultados, pode-se avaliar a hipótese definida no item (b), e rejeitar a hipótese nula de que a técnica $T4_{comp}$ não está formulada de maneira consistente, uma vez que a média de pontos atribuídos pelos participantes foi superior a 50%.

Em suma, a partir dos resultados observados, pode-se inferir que a técnica de leitura $T4_{comp}$apresentava consistência adequada, segundo a opinião dos participantes, e poderia ser usada como referência para a elaboração das demais técnicas.

Em relação ao formato de escrita da técnica pode-se perceber que, de acordo com a avaliação pessoal dos participantes, o formato texto foi ligeiramente melhor que o formato fluxo (cerca de 2% maior).No entanto, analisando-se os resultados do ponto de vista da efetividade, ou seja, dos acertos das marcações, pode-se perceber que ambos os formatos apresentaram resultados de efetividade similares em relação às etapas de aplicação da técnica. Assim, uma síntese geral é apresentada a seguir, com base na Tabela 5.3.

A partir da Tabela 5.3, o que se pode inferir e ressaltar em relação à efetividade associada aos formatos texto e fluxograma da técnica $T4_{comp}$ são os seguintes pontos:

- Em relação à Etapa I pode-se observar que os participantes foram mais efetivos no uso do formato fluxograma (20%) do que no formato texto (14%);
- Em relação à Etapa II pode-se observar que os participantes foram mais efetivos no uso do formato texto (41%) do que no formato fluxograma (30%); e
- Em relação à Etapa III, que é a etapa de comparação entre os artefatos, a identificação de defeitos em cada um dos artefatos ocorreu da seguinte forma: no artefato MEF os participantes foram mais efetivos no uso do formato texto (58%) do que no formato fluxograma (52%), e no REQ, os participantes foram mais efetivos no uso do formato fluxograma (40%) do que no formato texto (24%).

Assim, no geral, calculando-se a média de efetividade geral dos acertos de marcações em cada etapa, os dois formatos foram igualmente efetivos, apresentando assim média de 33%, o que sugeriu que qualquer um dos dois formatos poderia ser usado sem interferência na efetividade de aplicação da técnica.

Com base no formulário de viabilidade no qual foram coletadas opiniões e sugestões, os participantes ressaltaram os seguintes pontos:

- Recomendaram reforçar, durante o treinamento, que a técnica $T4_{comp}$ é composta de três etapas distintas – preparação de um dos artefatos; preparação do segundo artefato; e comparação entre os dois artefatos;
- Relataram que defeitos relacionados a requisitos derivados (requisitos que compartilham propriedades e estão associados via a associação de extensão com outro(s) requisito(s)) não eram capturados pelos estereótipos sendo, portanto, um fato importante que foi posteriormente adicionado à técnica $T4_{comp}$;
- Relataram problemas de dificuldade de entendimento durante a leitura da Etapa II da técnica, gerando também correções em uma versão posterior; e
- Relataram que o uso dos estereótipos era uma abordagem interessante e que facilitou o uso da técnica.

A partir dos resultados obtidos nesse estudo de viabilidade, foi possível estabelecer o Estudo de Viabilidade II que teve como principal objetivo, avaliar a efetividade e eficiência da técnica de leitura $T4_{comp}$ na detecção de defeitos. Esse estudo é apresentado resumidamente a seguir.

Tabela 5.3 - Efetividade no uso dos estereótipos em relação às Etapas I, II e III.

Formato usado	Efetividade	Etapa I (marcação no REQ)			Etapa II (marcação na MEF)			Etapa III (comparação entre os documentos)			
		Estereótipos usados para marcação do REQ			Estereótipos utilizados para marcação da MEF			Estereótipos utilizados para indicar os possíveis defeitos entre um documento e outro			
								Marcação de defeitos na MEF		Marcação de defeitos no REQ	
		EssReq	**TecReq**	**Syntax**	**SuperState**	**SubState**	**Syntax**	**SuperState ∩ IEEE Requirement Missing**	**SubState ∩ IEEE Requirement Missing**	**EssReq ∩ IEEE Design Missing**	**TecReq ∩ IEEE Design Missing**
Texto	(a) # marcações corretas feitas pelos participantes	2	52	2	5	44	2	8	14	5	48
	(b) # total de marcações corretas	65	156	39	13	65	13	13	26	65	117
	% (a/b)	**3%**	**33%**	**5%**	**38%**	**68%**	**15%**	**62%**	**54%**	**8%**	**41%**
	média percentual da Etapa	**14%**			**41%**			**58%**		**24%**	
Fluxograma	(c) # marcações corretas feitas pelos participantes	3	47	10	4	33	1	8	11	21	55
	(d) # total de marcações corretas	65	156	39	13	65	13	13	26	65	117
	%(c/d)	**5%**	**30%**	**26%**	**31%**	**51%**	**8%**	**62%**	**42%**	**32%**	**47%**
		20%			**30%**			**52%**		**40%**	

5.3 Estudo de Viabilidade II

Com base nos resultados encontrados no Estudo de Viabilidade I, e motivados pelo potencial da família de técnicas de leitura RTSS melhorias pontuais foram inseridas nas técnicas. Em particular, a técnica de leitura T4, foi novamente avaliada sobretudo com respeito aos pontos destacados a seguir:

- *efetividade* da técnica com respeito à detecção de defeitos;
- *eficiência* da técnica com respeito à detecção de defeitos; e
- se existe diferença entre os formatos estabelecidos (formato texto e formato fluxograma).

Assim, os resultados encontrados com o Estudo de Viabilidade II foram publicados na forma de um artigo no evento internacional **"*International Conference on Enterprise Information System* (ICEIS)"**com o título **"*Verification and Validation Activities for Embedded Systems: A Feasibility Study on a Reading Technique for SysML Models*"** (ANTONIO et.al, 2014).

A seguir será apresentada uma síntese das principais evidências encontradas e relatadas no artigo.

- A partir dos 26 participantes que realizaram o experimento, pode-se observar que pelo menos metade deles (50%) encontrou uma média de 72% dos defeitos em relação um oráculo previamente definido;
- Em se tratando do tempo, os participantes levaram em média 48 minutos para detectar pelo menos 72% dos defeitos;
- Como apresentado no estudo, verificou-se que não existe diferença significativa entre os formatos utilizados pela técnica de leitura (formato texto ou fluxograma); e
    - O uso de estereótipos para a marcação dos pares de artefatos SysML e comparação entre eles, se mostrou uma abordagem viável, uma vez que pelo menos 50% dos participantes detectaram pelo menos 70% dos defeitos.

5.4 Considerações Finais

A partir dos resultados encontrados nos estudos de viabilidade I e II foi possível refinar as considerações para as outras técnicas de leitura da família RTSS. Dentre os principais pontos utilizados, pode-se citar:

- O uso de estereótipos se mostrou uma alternativa viável à abordagem tradicional de marcação dos artefatos inspecionados;
- O *template* utilizado para formalizar as Diretrizes das técnicas de leitura se mostrou consistente para estruturar as outras técnicas de leitura da família RTSS;
- O uso do formato texto e ou de fluxograma, não acarreta impactos significativos na detecção de defeitos, não havendo, portanto diferenças em usar um formato ou outro;

Capítulo 6
EXEMPLOS DE APLICAÇÃODAS TÉCNICAS DE LEITURA

As técnicas RTSS foram estabelecidas para serem utilizadas, seja ao longo do processo SYSMOD ou quando houver outro processo que use os artefatos pertinentes às técnicas RTSS. Este capítulo apresenta exemplos de aplicação conduzidos para explorar o uso das técnicas RTSS nesses distintos momentos.

6.1 Considerações Iniciais

Considerando-se a proposição das técnicas RTSS (Capítulo 3) e as avaliações experimentais anteriormente apresentadas (Capítulo 4) este capítulo tem como objetivo mostrar três exemplos de aplicação que exploram a utilização das técnicas RTSS. Em particular, avaliaram-se as técnicas T1, T2, T3, T4 e T5 e como os defeitos identificados por uma determinada técnica foram (eventualmente) propagados para outras técnicas, considerando-se a ordem de aplicação disponível na Figura 3.2 (Capítulo 3) do processo SYSMOD adaptado. Assim, apresenta-se, passo a passo, a aplicação das técnicas em três exemplos, dos quais dois fizeram uso do processo SYSMOD e o outro, embora não tenha utilizado esse processo, foi desenvolvido com o suporte de artefatos da SysML.

Esses exemplos foram cuidadosamente selecionados para mostrar a aplicação das técnicas em contextos diferentes, de acordo com os seguintes Exemplos de aplicação:

(i) APE (*Active Phase Experiment*) – esse exemplo está relacionado com um sistema embarcado que representa uma aplicação VLT (*Very Large Telescope*). Esse sistema foi utilizado no experimento APE e foi uma sugestão dada pelo autor do livro (WEILKIENS, 2008), Tim Welkins, em contato oportuno realizado pelo autor deste trabalho. Esse exemplo explorou o uso das técnicas T1 e T2;
(ii) Computador de bordo – esse é um exemplo do sistema de computador de bordo utilizado como exemplo pelo autor Tim Welkins no livro em que ele

propõe o processo SYSMOD (WEILKIENS, 2008). Nesse exemplo foi avaliadas as técnicas T1, T2, T3 e T4 respectivamente; e finalmente
(iii) VANT Tiriba – esse exemplo trata da especificação de um Veículo Aéreo Não Tripulado utilizado em um trabalho de mestrado orientado por uma pesquisadora que participa do projeto INCT-SEC (SILVA, 2012). Nesse caso, foi avaliada a técnica T5.

Este capítulo está organizado da seguinte maneira: na Seção 6.2 apresentam-se os exemplos: APE; na Seção 6.3: Computador de Bordo; e finalmente na Seção 6.4 apresenta-se o exemplo da VANT Tiriba.

6.2 APE (*Active Phase Experiment*)

O projeto MBSE/APE[9] iniciado em 2004 é desenvolvido por um consórcio formado pela Agencia Especial Européia (ESO), Instituto de Astrofísica de Canárias (IAC), Observatório Astrofísico de *Arcetri*, Instituto *Nazionale d'Astrofisica* (INAF) e o *Laboratoire d'Astrophysique de Marseille* (LAM). O principal objetivo deste projeto é avaliar, integrar e validar quatro tecnologias diferentes de Óptica Adaptativa (OA) e de sensores ópticos visando minimizar os impactos de efeitos de "*tip-tilt*" e "*piston*" causados por fenômenos atmosféricos em espelhos primários de telescópios gigantes, i.e. *Very Large Telescope* (VLT).

Por ser um projeto extremamente complexo, os projetistas decidiram aplicar SysML para ter um gerenciamento de requisitos mais rigoroso. O projeto tem uma estrutura modular montado sobre uma mesa óptica. Os principais subsistemas são:

- Gerador de Turbulência (MAPS);
- Segmentador de Espelhos Ativos (da sigla em inglês, *Active Segmented Mirror* - ASM);
- Um interferômetro dual para o deslocamento de fase chamado de *Internal Metrology* (IM);
- Quatro sensores ópticos;
- Um quadro de junção; e
- Um *Opto-Mechanical Componentes* (OMB)

Esses elementos podem ser divididos em quatro grupos principais que são óptico, mecânico, eletrônico e interfaces de software.

Como dito anteriormente, esse exemplo foi sugerido pelo autor do livro (WEILKIENS, 2008), Tim Welkins, o qual foi desenvolvido por meio do processo SYSMOD. A seguir descreve-se a aplicação de cada uma das técnicas RTSS nesse exemplo, apresentando-se, passo a passo, como elas foram aplicadas.

a) Técnica de Leitura $T1_{reg}$ (REQ, IBD)

Como exposto no Capítulo 3, a Técnica $T1_{reg}$ explora a identificação de defeitos relativos a registradores conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

[9]http://mbse.gfse.de/

Etapa I – Preparação do REQ

Na primeira etapa da Técnica, são realizadas as devidas marcações com auxílio dos estereótipos «UL98CPUProg» e «UL98CPUReg», como indicado nas Diretrizes da Etapa 1 da Técnica $T1_{reg}$, conforme apresentado na Seção 4.2, Capítulo 4. A Figura 6.1 apresenta o diagrama REQ do exemplo APE após as marcações.

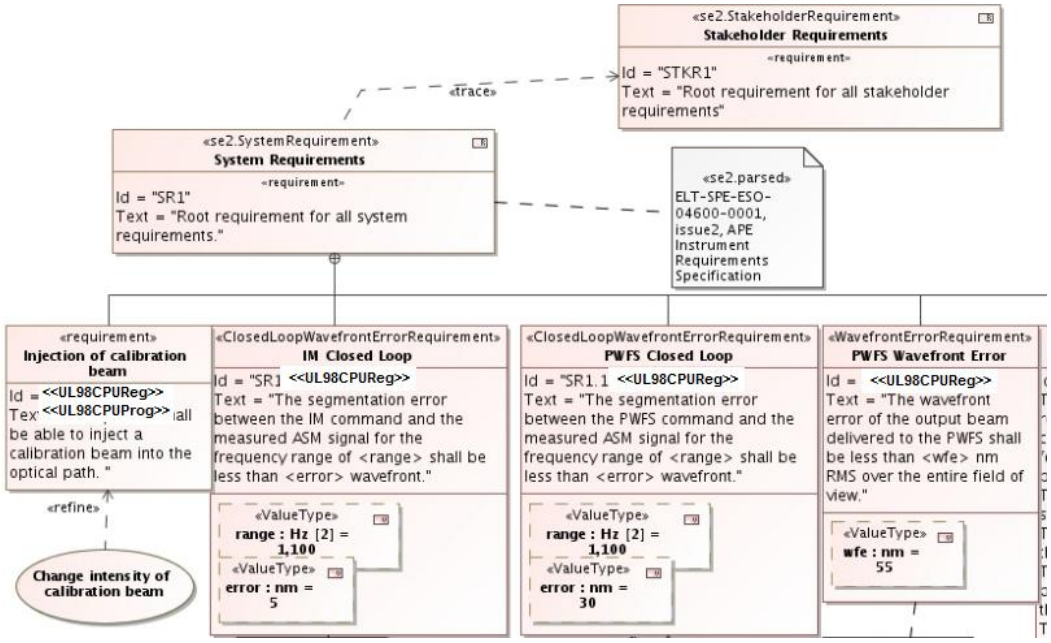


Figura 6.1 - Diagrama REQ (MBSE, 2011) adaptado após aplicação da Etapa I da Técnica $T1_{reg}$.

Como se pode verificar, o requisito "InjectionOfCalibrationBean" está marcado com os estereótipos «UL98CPUProg» e «UL98CPUReg», pois conforme as Diretrizes da técnica $T1_{reg}$ os requisitos que fizerem referência à inicialização e utilizarem alguma palavra-chave como, por exemplo, "luz" em que se deve marcá-la. De modo análogo, os requisitos "ImCloseLoop", "PWFSClosedLoop" e "PWFSWaveFrntError" foram marcados com o estereótipo «UL98CPUReg», pois retratam requisitos que em um sistema embarcado possam vir a ser modelados na forma de registradores. A evidência de que esses requisitos acabam se transformando em registradores é constatada a partir do uso do termo "error"nos requisitos que levam ao uso de um registrador de propósito geral em uma linguagem de mais baixo nível como.

Etapa II– Preparação do IBD

Na segunda etapa da técnica de leitura, são realizadas as devidas marcações com respeito ao uso dos estereótipos «UL98CPUProg» e «UL98CPUReg» no Diagrama IBD, como apresentado nas Diretrizes da Etapa 2 da Técnica $T1_{reg}$, Seção 4.2, Capítulo 4.

Durante a aplicação da técnica, foram identificados três blocos principais - "LightSourcesModule", "PhaseModulator" e "SignalDecoder" no modelo IBD. No entanto, como se verifica na Figura 6.2, não foi feita nenhuma marcação no diagrama IBD, pois não foi possível identificar palavras-chave como, por exemplo, "temperatura", "inicialização",

etc., que poderiam ser representadas por meio de registradores de propósito gerais em linguagens de mais baixo nível.

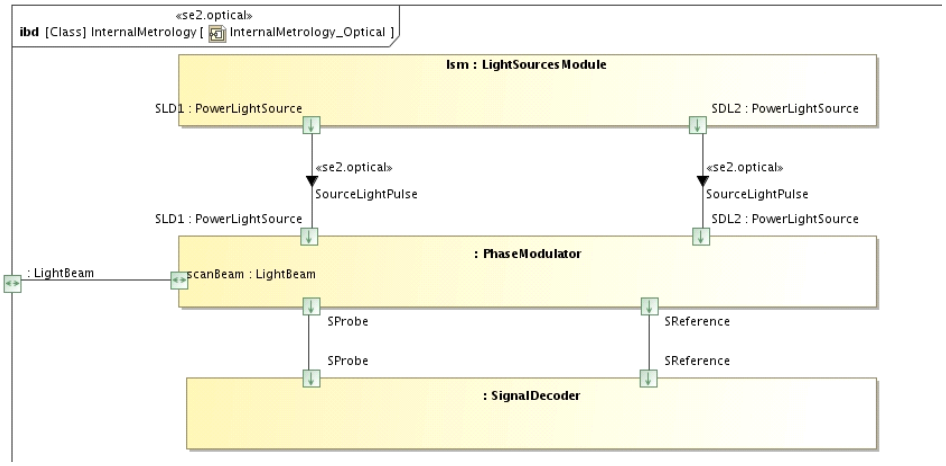


Figura 6.2 - Diagrama IBD (MBSE, 2011) após aplicação da Etapa II da técnica $T1_{reg}$.

Etapa III– Comparação entre os diagramas REQ e IBD

Na terceira etapa da técnica de leitura, são realizadas as verificações comparando-se os Diagramas REQ e IBD com respeito aos estereótipos «UL98CPUProg» e «UL98CPUReg» previamente anotados em ambos os diagramas durante as Etapas I e II.

Considerando-se o primeiro passo da Etapa III, que pede para verificar se os requisitos marcados no diagrama REQ com os estereótipos«UL98CPUProg»e«UL98CPUReg» estão consistentes com as marcações feitas noIBD, verificou-se que o requisito "InjectionOfCalibrationBean" possuía essas marcações e elas não estavam consistentes com o diagrama IBD. Tal inconsistência foi identificada a partir das Diretrizes C.1 e D.1 da Etapa III fazendo com que o requisito "InjectionOfCalibrationBean" fosse marcado no diagrama REQ com o estereótipo «IEEEDesignMissing».

Considerando-se o segundo passo da Etapa III, que pede para comparar as marcações do diagrama IBD com relação ao diagrama REQ, pelo fato do diagrama IBD não ter recebido anotações (vide Figura 6.2), nenhum defeito foi assinalado.

Além disso, à medida que os defeitos foram identificados, o formulário de discrepância foi preenchido conforme as Diretrizes C.2 e D.2 da técnica $T1_{reg}$. Assim, a Tabela 6.1descreve parte do Formulário de Discrepâncias, mostrando os defeitos que foram detectados pela técnica $T1_{reg}$.

Tabela 6.1 - Parte do Formulário de Discrepância após aplicação da técnica $T1_{reg}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito	Severidade
1	**REQ**	«IEEEDesignMissing»	**Registrador**	InjectionOfCalibrationBean	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Registrador**	IMClosedLoop	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Registrador**	PWFSClosedLoop	«IEEECriticalSeverity»
4	**REQ**	«IEEEDesignMissing»	**Registrador**	PWFSWaveFrontError	«IEEECriticalSeverity»

Após a aplicação da técnica $T1_{reg}$, foram observados 4 defeitos que correspondem ao conceito de registrador. Como consequência disso a modelagem no diagrama IBD estava incompleta em relação ao conceito de registrador relativo à norma de certificação UL-98. Em particular, essa ausência de informação pode estar associada, como por exemplo, a uma falta de entendimento do projetista do SE em relação à especificação do REQ e que culminou na omissão de informação no diagrama IBD. Desse modo, os defeitos foram classificados como sendo «IEEEDesignMissing», com alta severidade «IEEECriticalSeverity» uma vez que tais requisitos são essenciais para o funcionamento do sistema APE pois envolvem a calibração e inicialização dos equipamentos.

Em suma, caso esses defeitos não sejam corrigidos já nos momentos iniciais da especificação/modelagem do processo SYSMOD, tais defeitos podem se propagar para outras fases de desenvolvimento chegando até a fase de implementação da aplicação, levando, eventualmente, a uma implementação errônea dos requisitos necessários. Em contrapartida, caso a Técnica$T1_{reg}$ tivesse sido utilizada, esses defeitos teriam sido sinalizados assim que os diagramas REQ e IBD estivessem concluídos, evitando retrabalho futuro.

b) Técnica de Leitura $T1_{int}$ (REQ, IBD)

Como exposto no Capítulo 3, a Técnica $T1_{int}$ explora a identificação de defeitos relativos a interrupções conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa Técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizados as marcações com auxílio dos estereótipos «UL98Interrupt» e «UL98Clock», como indicado nas Diretrizes da Etapa 1 da Técnica $T1_{int}$, conforme apresentado na descrição da Técnica $T1_{int}$. A Figura 6.3 ilustra o diagrama REQ após essas marcações.

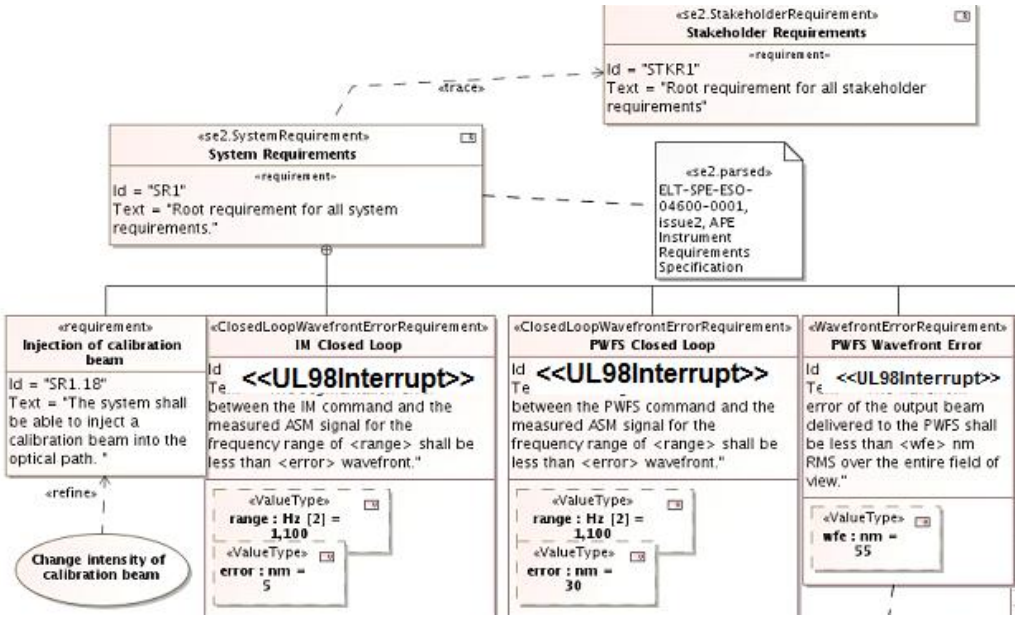


Figura 6.3 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da Técnica $T1_{int}$.

Conforme descrito nas Diretrizes da Técnica de Leitura, os blocos "IMClosedLoop" e "PWFSClosedLoop" indicam o uso do termo frequência que associa-se a uma interrupção de mais baixo nível. Além disso, o requisito "PWFSWaveFrontError" também sugere o uso de interrupções pois faz o tratamento de erro para o comprimento de onda em nanômetros assim o estereótipo UL98Interrupt foi marcado nesse bloco do IBD.

Etapa II – Preparação do IBD

Na segunda etapa da Técnica de leitura, são realizadas as devidas marcações com respeito ao uso dos estereótipos «UL98Interrupt» e «UL98Clock»no diagrama IBD, conforme apresentado na descrição da Técnica $T1_{int}$.

Como pode se verificar na Figura 6.4, foi feita uma marcação com o estereótipo UL98Interrupt no bloco "PhaseModulator" uma vez que esse bloco trata de uma comunicação serial bem como o uso de uma modulação do comprimento de onda, o que pode indicar internamente pelo uso de uma interrupção de mais baixo nível "SReference".

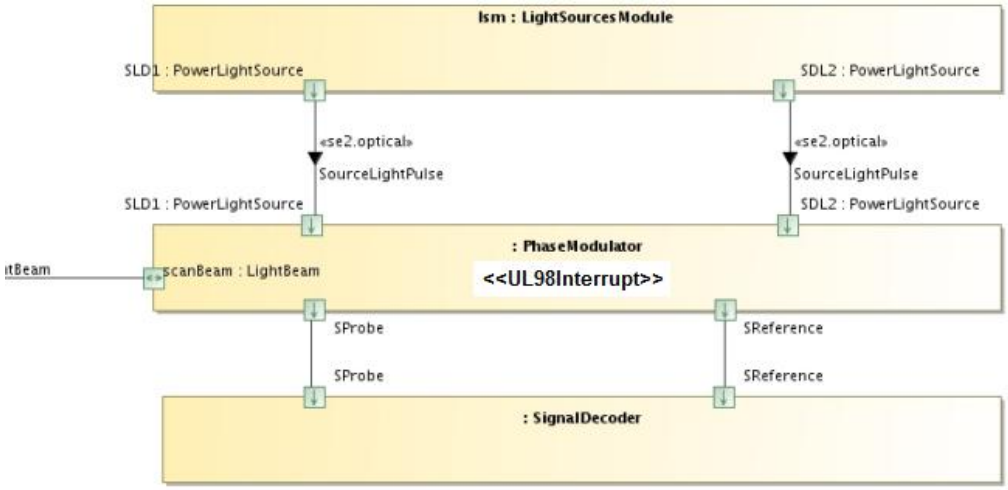


Figura 6.4 – Diagrama IBD(MBSE, 2011) após aplicação da Etapa II da técnica $T1_{int}$.

Etapa III – Comparação entre os diagramas REQ e IBD

Na terceira etapa da Técnica de Leitura, são realizadas as verificações comparando-se os Diagramas REQ e IBD simultaneamente com respeito aos estereótipos «UL98Interrupt» e «UL98Clock» previamente anotados.

Considerando-se o primeiro passo da Etapa III, que pede para verificar se os requisitos marcados no diagrama REQ com os estereótiposUL98Interrupte UL98Clock estão consistentes com as marcações feitas no IBD, verificou-se que os requisitos "PWFSClosedLoop" e "PWFSWaveFrntError" possuíam essas marcações e elas não estavam consistentes com o diagrama IBD. Tais inconsistências foram identificadas a partir das Diretrizes C.1 e C.1 da Etapa III fazendo com que esses requisitos fossem marcados no diagrama REQ com o estereótipo «IEEEDesignMissing». De modo análogo, para comparar as marcações do diagrama IBD com as marcações do diagrama REQ, pode-se verificar que o bloco "PhaseModulator" não possuía um requisito correspondente. Dessa forma, tal bloco do diagrama IBD foi marcado com o estereótipo «IEEERequirementMissing».

Além disso, à medida que os defeitos foram identificados, o formulário de discrepância foi preenchido conforme as Diretrizes C.2, D.2, E.2 e F.2 da Técnica $T1_{int}$. Assim, a Tabela 6.2 apresenta os defeitos detectados e reportados nesse formulário.

Tabela 6.2 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{int}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Interrupção**	PWFSClosedLoop	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Interrupção**	PWFSWaveFrontError	«IEEECriticalSeverity»
3	**IBD**	«IEEERequirement Missing»	**Interrupção**	PhaseModulator	«IEEECriticalSeverity»

Após a aplicação da Técnica $T1_{int}$, foram observados 3 defeitos que correspondem ao conceito de interrupção. Como consequência disso a especificação de requisitos no diagrama REQ e a modelagem no diagrama IBD tornaram-se inconsistentes em relação ao conceito de interrupção da norma de certificação UL-98. Desse modo, os defeitos foram classificados como sendo «IEEEDesignMissing»e «IEEERequirementMissing», ambos com alta severidade «IEEECriticalSeverity»uma vez que tais requisitos são essenciais para o funcionamento do sistema APE/VLT pois envolvem por exemplo a modulação em frequência de sinais ópticos.

c) Técnica de Leitura $T1_{mem}$ (REQ, IBD)

Como exposto no Capítulo 3, a Técnica $T1_{mem}$ explora a identificação de defeitos relativos a memória conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa da Técnica, são realizados as marcações com os estereótipos «UL98ExternalMemory», «UL98VolatileMemory»e«UL98ProtectMemory», Etapa 1 da Técnica $T1_{mem}$. A Figura 6.5 6.5 mostra o diagrama REQ do exemplo APE após as marcações.

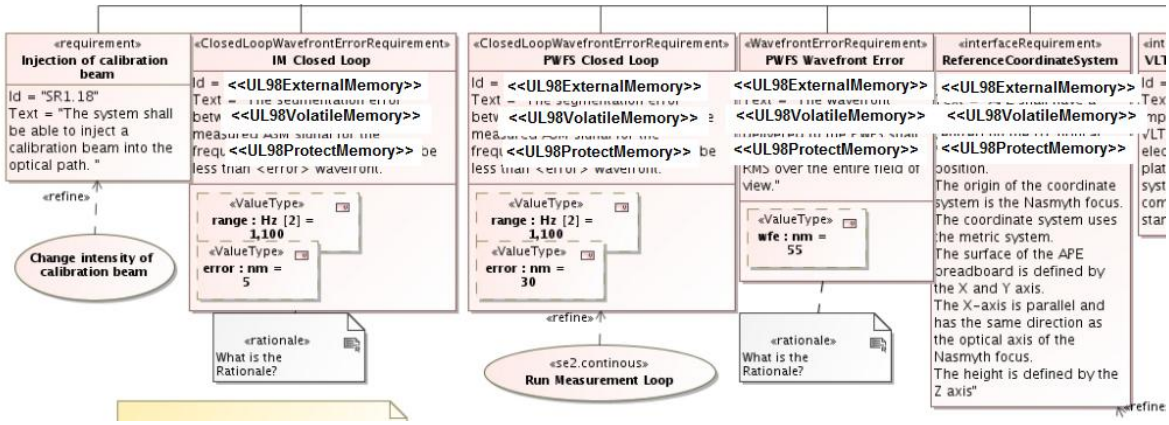


Figura 6.5 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da Técnica $T1_{mem}$.

Como se pode verificar, por exemplo, o requisito “IMClosedLoop” está marcado com os estereótipos «UL98ExternalMemory», «UL98VolatileMemory»e«UL98ProtectMemory», pois conforme as Diretrizes da Técnica $T1_{mem}$ os requisitos que fizerem referência à alguma palavra-chave como memória (estática, dinâmica) devem ser marcados. Em uma análise atenta, pode-se verificar, por exemplo, que o requisito "ReferenceCoordinateSystem" que manipula as variáveis/coordenadas (x;y;z) levando portanto a marcação do estereótipo no requisito em questão.

Etapa II – Preparação do IBD

Na segunda etapa da Técnica de Leitura, são realizados as devidas marcações com respeito ao uso dos estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory» no Diagrama IBD, como apresentado nas Diretrizes da Etapa 2 da Técnica $T1_{mem}$.

Durante a aplicação da técnica, foram identificados três blocos principais — "LightSourcesModule", "PhaseModulator" e "SignalDecoder" no modelo IBD que receberam a marcação dos estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory». Etapa 2 da Técnica $T1_{mem}$, conforme apresentado.A Figura 6.6 – Diagrama IBD(MBSE, 2011) após aplicação da Etapa II da Técnica T1mem.

ilustra parcialmente o diagrama IBD utilizado com a marcação do estereótipo.

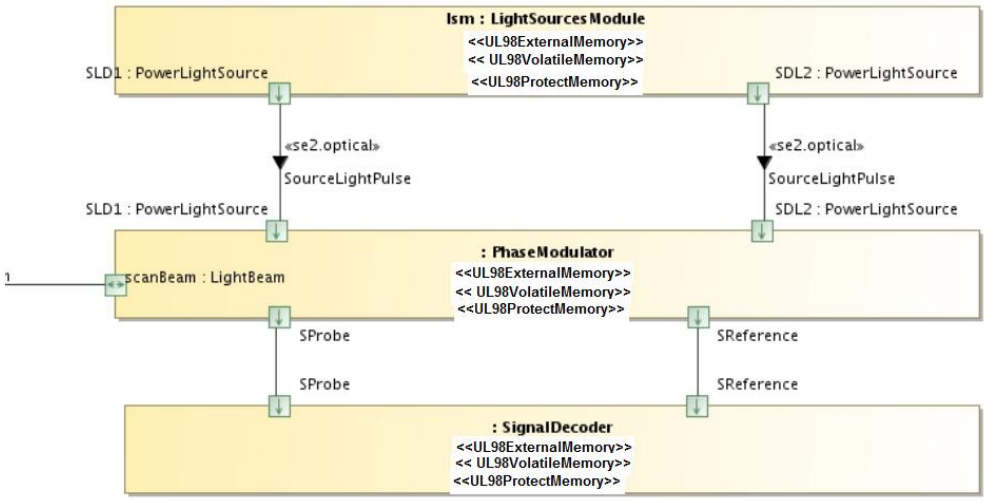


Figura 6.6 – Diagrama IBD(MBSE, 2011) após aplicação da Etapa II da Técnica $T1_{mem}$.

Etapa III – Comparação entre os diagramas REQ e IBD

Na terceira etapa, são realizadas as verificações comparando-se os Diagramas REQ e IBD simultaneamente com respeito aos estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory» previamente anotados.

Considerando-se o primeiro passo da Etapa III, que pede para verificar se os requisitos marcados no diagramaREQ com os estereótipos mencionados acimaestão consistentes com as marcações feitas no IBD, verificou-se que os requisitos "PWFSClosedLoop", "PWFSWavefrontError" e "ReferenceCoordinateSystem" possuíam essas marcações e elas não estavam consistentes com o diagrama IBD. Tais inconsistências foram identificadas a partir das Diretrizes A.1, B.1, C.1, D.1, E.1 e F.1 da Etapa III fazendo com que esses requisitos fossem marcados no diagrama REQ com o estereótipo «IEEEDesignMissing». De modo análogo, para comparar as marcações do diagrama IBD com as marcações do diagrama REQ, pode-se verificar que o bloco "IMClosedLoop" não possuía um requisito correspondente. Dessa forma, tal bloco do diagrama IBD foi marcado com o estereótipo «IEEERequirementMissing».

Além disso, à medida que os defeitos foram identificados, o formulário de discrepância foi preenchido conforme as Diretrizes A.2, B.2, C.2, D.2, E.2 e F.2 da Técnica $T1_{mem}$. Assim, a Tabela 6.3 apresenta os defeitos detectados e reportados nesse formulário.

Tabela 6.3 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{mem}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Memória**	PWFSClosedLoop	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Memória**	PWFSWaveFrontError	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Memória**	ReferenceCoordinateSystem	«IEEECriticalSeverity»
4	**IBD**	«IEEERequirementMissing»	**Memória**	IMClosedLoop	«IEEECriticalSeverity»

A partir da Tabela 6.3 pode-se observar que foram detectados 4 defeitos que correspondem ao conceito de memória. Como consequência dissoa especificação de requisitos no diagrama REQ e a modelagem no diagrama IBD tornaram-se inconsistentes em relação aoconceito de memória da norma de certificação UL-98. Desse modo, os defeitos foram classificados como sendo «IEEEDesignMissing»e «IEEERequirementMissing», ambos com alta severidade «IEEECriticalSeverity»uma vez que tais requisitos são essenciais para o funcionamento do sistema APE. Como exemplo, o terceiro defeito (Tabela 6.3), é um defeito que deve ser mitigado, pois o requisito em questão envolve o uso de coordenadas necessárias para o posicionamento das lentes do VLT.

d) Técnica de Leitura $T1_{in}$ (REQ, IBD)

Como exposto no Capítulo 3, a Técnica $T1_{in}$ explora a identificação de defeitos relativos a entrada conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa, são realizadas as marcações com os estereótipos «UL98PlausibilityCheck», como indicado nas Diretrizes da Etapa 1 da Técnica $T1_{in}$ conforme apresentado na descrição rigorosa da técnica $T1_{in}$. A Figura 6.7ilustra os estereótipos anotados no diagrama REQ após a aplicação da técnica.

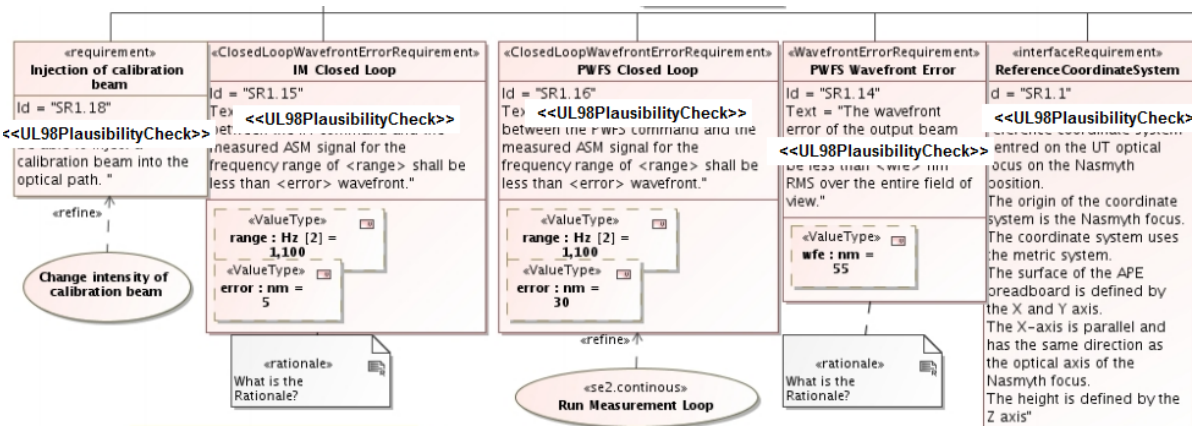


Figura 6.7 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da Técnica $T1_{in}$.

A partir da aplicação da Técnica de Leitura, foi possível identificar os requisitos que retratam o uso de elementos como entrada de dados. Como exemplo, o requisito "InjectionOfCalibrationBean", que como pode ser observado, requer que a intensidade do feixe de luz do laser seja alterada/calibrada, como consequência, tal calibração deve ocorrer por meio de um dispositivo de entrada.

Etapa II – Preparação do IBD

Na segunda etapa da técnica, são realizadas as marcações com o estereótipo «UL98PlausibilityCheck» no diagrama IBD, conforme apresentado na descrição da Técnica $T1_{in}$.

Considerando-se as Diretrizes da Técnica, foram feitas marcações nos blocos "PhaseModulator" e "SignalDecoder" uma vez que tais blocos recebem como entrada o feixe do laser como entrada.

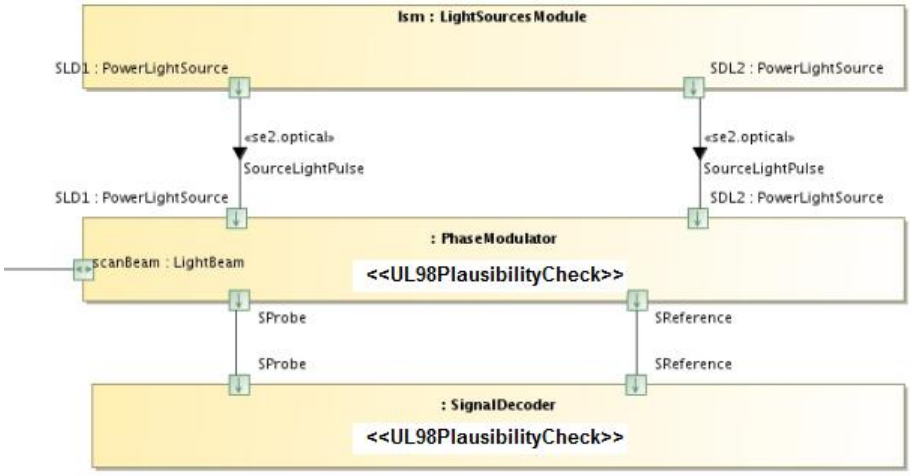


Figura 6.8- Diagrama IBD(MBSE, 2011) após aplicação da Etapa II da Técnica $T1_{in}$.

Etapa III – Comparação entre os diagramas REQ e IBD

Na terceira etapa da técnica de leitura, são realizadas as verificações comparando-se os Diagramas REQ e IBD simultaneamente com respeito ao estereótipo «UL98PlausibilityCheck» previamente anotados.

Considerando-se o primeiro passo da Etapa III, que pede para verificar se os requisitos marcados no diagramaREQ com o estereótipo «UL98PlausibilityCheck» estão consistentes com as marcações feitas no IBD, verificou-se que os requisitos "InjectionOfCalibrationBean", “PWFSClosedLoop”, “PWFSWaveFrontError" e "ReferenceCoordinateSystem" possuíam essas marcações e elas não estavam consistentes com o diagrama IBD. Tais inconsistências foram identificadas a partir das Diretrizes A.1 da Etapa III fazendo com que esses requisitos fossem marcados no diagrama REQ com o estereótipo «IEEEDesignMissing». De modo análogo, para comparar as marcações do diagrama IBD com as marcações do diagrama REQ, pode-se verificar que o bloco "IMClosedLoop" não possuía um requisito correspondente. Dessa forma, tal bloco do diagrama IBD foi marcado com o estereótipo «IEEERequirementMissing». Contudo, não foram encontradas anomalias sintáticas como descrito na Diretriz C.1 da Técnica $T1_{in}$.

Além disso, à medida que os defeitos foram identificados, o formulário de discrepância foi preenchido conforme as Diretrizes A.2 e B.2 da Técnica $T1_{in}$. Assim, a Tabela 6.4 apresenta os defeitos detectados e reportados nesse formulário.

Tabela 6.4 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{in}$

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Entrada**	InjectionOfCalibrationBean	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Entrada**	IMClosedLoop	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Entrada**	PWFSClosedLoop	«IEEECriticalSeverity»
4	**REQ**	«IEEEDesignMissing»	**Entrada**	ReferenceCoordinateSystem	«IEEECriticalSeverity»
5	**IBD**	«IEEERequirementMissing»	**Entrada**	PWFSWavefrontError	«IEEECriticalSeverity»

Após a aplicação da Técnica $T1_{in}$, foram observados 5 defeitos que correspondem ao conceito de entrada de dados. Como consequência disso a especificação de requisitos no diagrama REQ e a modelagem no diagrama IBD tornaram-se inconsistentes em relação ao conceito de entrada de dados da norma de certificação UL-98. Desse modo, os defeitos foram classificados como sendo «IEEEDesignMissing»e «IEEERequirementMissing», ambos com alta severidade «IEEECriticalSeverity» uma vez que tais requisitos são essenciais para o funcionamento do sistema APE. Assim, por exemplo, o quarto defeito (Tabela 6.4), envolve o controle de erro no modelo IBD, e que muito provavelmente, será transcrito em mais baixo nível como uma estrutura condicional, pois caso um erro de execução ocorra, o sistema APE/VLT deve ser capaz de contornar esse tipo mau funcionamento.

Em suma, caso o projetista do VLT deseje mitigar esse tipo de defeito e evitar retrabalho futuro, é importante que a Técnica $T1_{in}$seja utilizada, sobretudo para se reduzir o impacto na qualidade final do produto gerado.

e) Técnica de Leitura $T1_{out}$ (REQ, IBD)

Como exposto no Capítulo 3, a Técnica $T1_{out}$ explora a identificação de defeitos relativos à saída de dos conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa, são realizadas as marcações com os estereótipos «UL98PlausibilityCheck», como indicado nas Diretrizes da Etapa 1 da Técnica $T1_{out}$. A Figura 4.9 ilustra o estereótipo anotado no diagrama REQ após a aplicação da técnica.

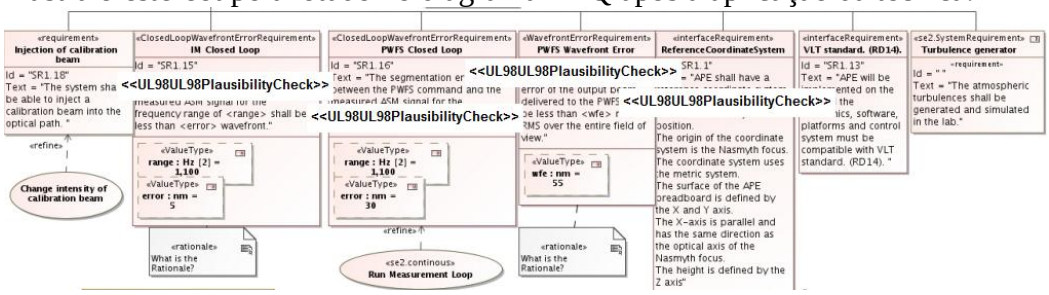


Figura 6.9 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da Técnica $T1_{out}$.

A partir da aplicação da técnica de leitura, foi possível identificar os requisitos que retratam o uso de elementos como entrada de dados. Como exemplo, o requisito

"IMClosedLoop", que como pode ser observado, requer que a frequência de luz seja de 1,1 KHz e que erro seja menor que 5 nanômetros. Nesse caso, considerando-se que tal requisito poderia ser representado como um elemento de mais baixo nível, como por exemplo, um bloco em Simulink, com pelo menos duas saídas requeridas (uma para a frequência e outra para o erro), tal requisito foi marcado com o estereótipo «UL98PlausibilityCheck».

Etapa II – Preparação do IBD

Na segunda etapa da técnica, são realizados as marcações com o estereótipo «UL98PlausibilityCheck» no diagrama IBD, como apresentado nas Diretrizes da Etapa 2 da Técnica $T1_{out}$, considerando-se a Diretrizes da Técnica, foram feitas marcações nos blocos "LighSourcesModule" e "PhaseModulator" uma vez que tais blocos realizam um pré-processamento e retornar um valor utilizado pelo APE.

Na segunda etapa da Técnica de Leitura, são realizados as devidas marcações com respeito ao uso dos estereótipos «UL98PlausibilityCheck» no Diagrama IBD, como apresentado nas Diretrizes da Etapa 2 da Técnica $T1_{out}$. A Figura 6.10, retrata os blocos e as marcações realizadas.

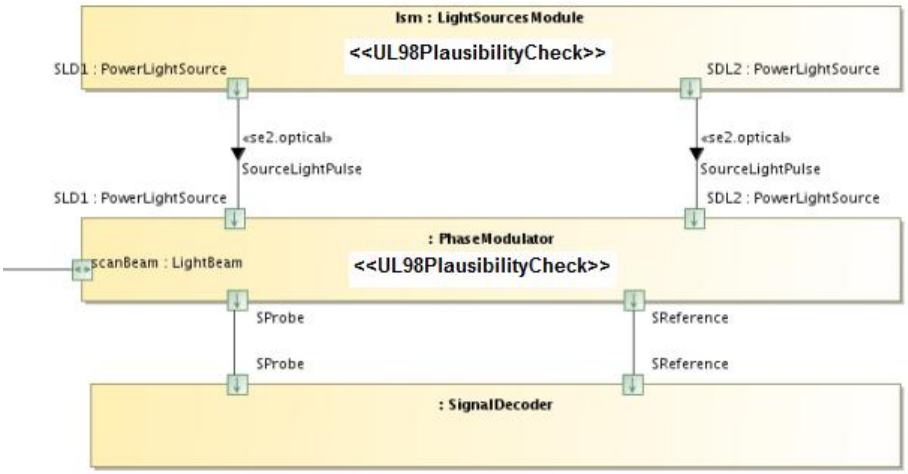


Figura 6.10 – Diagrama IBD (MBSE, 2011) após aplicação da Etapa II da Técnica $T1_{out}$.

Etapa III – Comparação entre os diagramas REQ e IBD

Na terceira etapa da Técnica de Leitura, são realizadas as verificações comparando-se os Diagramas REQ e IBD simultaneamente com respeito ao estereótipo «UL98PlausibilityCheck» previamente anotado.

Considerando-se o primeiro passo da Etapa III, que pede para verificar se os requisitos marcados no diagrama REQ com o estereótipo «UL98PlausibilityCheck» estão consistentes com as marcações feitas no IBD, verificou-se que os requisitos "PWFSClosedLoop", "PWFSWaveFrontError" e "ReferenceCoordinateSystem" possuíam essas marcações e elas não estavam consistentes com o diagrama IBD. Tais inconsistências foram identificadas a partir das Diretrizes A.1 da Etapa III fazendo com que esses requisitos fossem marcados no diagrama REQ com o estereótipo «IEEEDesignMissing». De modo análogo, para comparar as marcações do diagrama IBD com as marcações do diagrama REQ, pode-se verificar que o bloco "IMClosedLoop" não possuía um requisito correspondente. Dessa forma, tal bloco do diagrama IBD foi marcado com o estereótipo «IEEERequirementMissing». Contudo, não foram encontradas anomalias sintáticas como descrito na Diretriz C.1 da Técnica $T1_{out}$.

Além disso, à medida que os defeitos foram identificados, o formulário de discrepância foi preenchido conforme as Diretrizes A.2 e B.2 da Técnica $T1_{out}$. Assim, a Tabela 6.5 apresenta os defeitos detectados e reportados nesse formulário.

Tabela 6.5 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{out}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Saída**	IMClosedLoop	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Saída**	PWFSClosedLoop	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Saída**	ReferenceCoordinate	«IEEECriticalSeverity»
4	**IBD**	«IEEERequirementMissing»	**Saída**	PWFSWavefrontError	«IEEECriticalSeverity»

Após a aplicação da Técnica $T1_{out}$, foram observados 4 defeitos dos quais três deles foram classificados como «IEEEDesignMissing» e um deles foi classificado como «IEEERequirementMissing». Como consequência disso, a especificação de requisitos no diagrama REQ e a modelagem no diagrama IBD tornaram-se inconsistentes em relação aoconceito de entrada de dados da norma de certificação UL-98. Desse modo, os defeitos foram classificados como alta severidade «IEEECriticalSeverity»uma vez que tais requisitos são essenciais para o funcionamento do sistema APE. Assim, por exemplo, o primeiro defeito (Tabela 6.5), envolve a especificação do dispositivo de metrologia (IM) que está ausente no modelo IBD. Essa omissão de informação pode levar a um mau funcionamento do sistema, em especial, para o ajuste fino das lentes do sistema VLT, pois como ser observado no requisito, é necessário que a frequência do feixe de luz do interferômetro, seja restrita a 1,1 KHz e com um erro de no máximo 5 nanômetros.

Em suma, caso o projetista do VLT deseje minimizar o impacto na reconstrução do VLT/APE, sobretudo com relação aos defeitos encontrados, é importante considerar o uso da Técnica $T1_{out}$, uma vez que ela detecta tais defeitos já nas fases iniciais de desenvolvimento.

f) Técnica de Leitura $T2_{reg}$ (REQ, BDD)

Como exposto no Capítulo 3, a Técnica $T2_{reg}$ explora a identificação de defeitos relativos a registradores conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «UL98CPUProg» e «UL98CPUReg», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{reg}$. A Figura 6.11 apresenta o diagrama REQ do exemplo APE após as marcações.

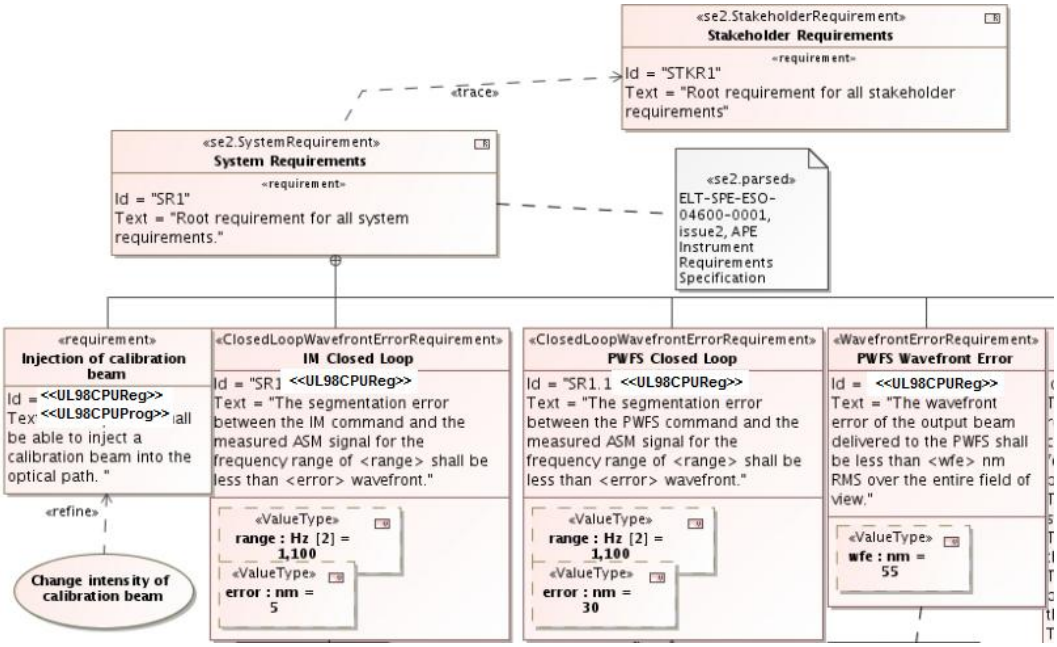


Figura 6.11 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da Técnica $T2_{reg}$.

Etapa II – Preparação do BDD

Na segunda etapa da técnica de leitura, são realizadas as devidas marcações com respeito ao uso dos estereótipos «UL98CPUProg» e «UL98CPUReg» no Diagrama BDD, como apresentado nas Diretrizes da Etapa 2 da Técnica $T2_{reg}$. A Figura 6.12 retrata as marcações realizadas.

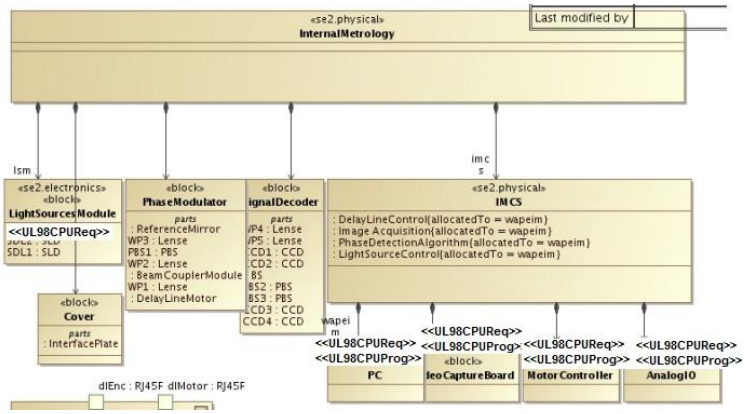


Figura 6.12 - Diagrama BDD (MBSE, 2011) após aplicação da Etapa II da Técnica $T2_{reg}$.

Etapa III – Comparação do REQ com BDD

Na terceira etapa da técnica de leitura, são realizadas as verificações comparando-se os Diagramas REQ e BDD com respeito aos estereótipos «UL98CPUProg» e «UL98CPUReg» previamente anotados em ambos os diagramas durante as Etapas I e II.

Considerando-se o primeiro e o segundo passo da Etapa III os seguintes defeitos foram detectados e parte do Formulário de Discrepâncias é apresentada na Tabela 6.6 com os defeitos detectados pela Técnica $T2_{reg}$.

Tabela 6.6 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{reg}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Registrador**	IMClosedLoop	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Registrador**	PWFSClosedLoop	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Registrador**	PWFSWaveFrontError	«IEEECriticalSeverity»
4	**REQ**	«IEEEDesignMissing»	**Registrador**	ReferenceCoordinateSystem	«IEEECriticalSeverity»
5	**REQ**	«IEEEDesignMissing»	**Registrador**	LightSourceModule	«IEEECriticalSeverity»
6	**BDD**	«IEEEReqMissing»	**Registrador**	PC	«IEEECriticalSeverity»
7	**BDD**	«IEEEReqMissing»	**Registrador**	VideoCaptureBoard	«IEEECriticalSeverity»
8	**BDD**	«IEEEReqMissing»	**Registrador**	MotorController	«IEEECriticalSeverity»
9	**BDD**	«IEEEReqMissing»	**Registrador**	AnalogIO	«IEEECriticalSeverity»

Após a aplicação da Técnica $T2_{reg}$, foram detectados9 defeitos que correspondem ao conceito de registrador. Como consequência, o diagrama BDD tornou-se inconsistente em relação ao conceito de registrador conforme a cobertura exigida pela norma de certificação UL-98. Desse modo, dos 9 defeitos encontrados, 5 deles foram classificados como sendo «IEEEDesignMissing» e 4 deles como sendo «IEEEReqMissing». Além disso, tais defeitos foram indicados como sendo de alta severidade «IEEECriticalSeverity»uma vez que tais requisitos/blocos são essenciais para o funcionamento do APE.

Vale ressaltar, que os defeitos 1, 2 e 3 detectados pela técnica $T2_{reg}$(vide Tabela 6.6), também foram detectados pela técnica $T1_{reg}$, o que sugere que tais defeitos foram propagados da fase de "Requisitos" para a fase de "Conhecimento de Domínio" do processo SYSMOD. Assim, como os defeitos não foram mitigados previamente (i.e., com o uso da técnica $T1_{reg}$), eles foram inseridos também no diagrama BDD.

g) Técnica de Leitura $T2_{int}$ (REQ, BDD)

Como exposto no Capítulo 3, a Técnica de Leitura $T2_{int}$ explora a identificação de defeitos relativos a interrupções conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizados as marcações com auxílio dos estereótipos «UL98Interrupt» e «UL98Clock», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{int}$, conforme apresentado na descrição da técnica $T2_{int}$. A Figura 6.13 ilustra o diagrama REQ após essas marcações.

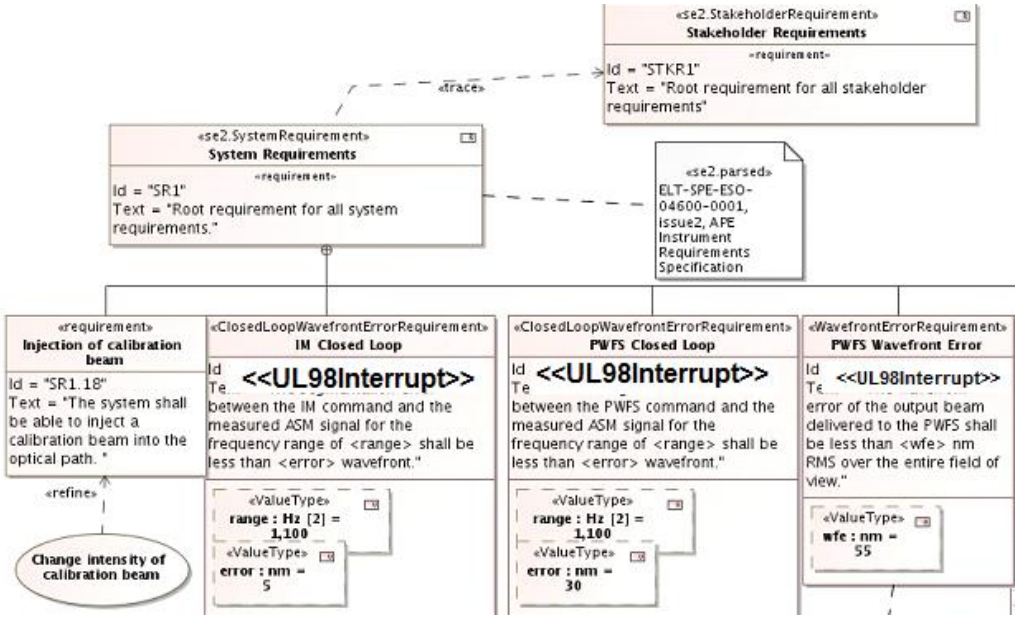


Figura 6.13 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da Técnica $T2_{int}$.

Etapa II – Preparação do BDD

Na segunda etapa da Técnica de Leitura, são realizados as devidas marcações com respeito ao uso dos estereótipos «UL98Interrupt» e «UL98Clock»no Diagrama BDD. A Figura 6.14, retrata os blocos e as marcações realizadas no BDD.

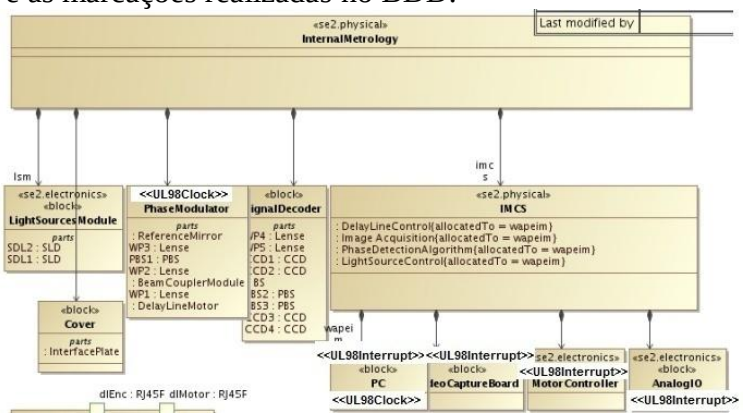


Figura 6.14 – Diagrama BDD (MBSE, 2011) após aplicação da Etapa II da Técnica $T1_{int}$.

Etapa III – Comparação entre os diagramas REQ eBDD

Após a aplicação dos passos anteriores, a partir do primeiro passo da Etapa III, que pede para verificar se os requisitos marcados no diagrama REQ com os estereótipos«UL98Interrupt»e«UL98Clock» estão consistentes com as marcações feitas noBDD, verificou-se que os requisitos “IMClosedLoop", "PWFSClosedLoop" e“PWFSWaveFrontError" que possuíam essas marcações não estavam consistentes com o diagrama BDD. Tais inconsistências foram identificadas a partir das Diretrizes A.1 e B.1 da Etapa III fazendo com que esses requisitos fossem marcados no diagrama REQ com o estereótipo «IEEEDesignMissing». De modo análogo, para comparar as marcações do diagrama BDD com as marcações do diagrama REQ, pode-se verificar que o bloco "PhaseModulator", "PC", "VideoCaptureBoard", "MotorController", "AnalogIO" não possuíam um requisito correspondente. Dessa forma, tais blocos do diagrama BDD foram marcados com o estereótipo «IEEERequirementMissing».

Assim, os seguintes defeitos foram detectados e parte do Formulário de Discrepâncias é resumido na Tabela 6.7 com os defeitos detectados pela Técnica $T2_{int}$.

Tabela 6.7 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{int}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Interrupção**	IMClosedLoop	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Interrupção**	PWFSClosedLoop	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Interrupção**	PWFSWaveFrontError	«IEEECriticalSeverity»
4	**BDD**	«IEEEDesignMissing»	**Interrupção**	PhaseModulator	«IEEECriticalSeverity»
5	**BDD**	«IEEEReqMissing»	**Interrupção**	PC	«IEEECriticalSeverity»
6	**BDD**	«IEEEReqMissing»	**Interrupção**	VideoCaptureBoard	«IEEECriticalSeverity»
7	**BDD**	«IEEEReqMissing»	**Interrupção**	MotorController	«IEEECriticalSeverity»
8	**BDD**	«IEEEReqMissing»	**Interrupção**	AnalogIO	«IEEECriticalSeverity»

Após a aplicação da técnica $T2_{int}$, foram detectados8dos quais 3 são defeitos relativos ao diagrama de requisitos (REQ) e 5 deles são relativos ao diagrama BDD. Assim, o diagrama BDD tornou-se inconsistente em relação ao conceito de interrupção conforme a cobertura exigida pela norma de certificação UL-98. Além disso, tais defeitos foram indicados como sendo de alta severidade «IEEEHighPriority» uma vez que tais requisitos/blocos são essenciais para o funcionamento do APE.

Vale ressaltar, que os defeitos 1, 2 e 3 detectados pela técnica $T2_{int}$(vide Tabela 6.7), também foram detectados pela técnica $T1_{int}$, o que sugere que tais defeitos foram propagados da fase de "Requisitos" para a fase de "Conhecimento de Domínio" do processo SYSMOD. Assim, como os defeitos não foram mitigados previamente (i.e., com o uso da técnica $T1_{int}$), eles foram inseridos (involuntariamente ou não) também no diagrama BDD.

Em relação aos defeitos de 4 a 8 pode-se inferir, que a modelagem do diagrama BDD é carente, sobretudo de especificação consistente que envolva segundo o ponto de vista da norma UL98.

h) Técnica de Leitura $T2_{mem}$ (REQ, BDD)

Como exposto no Capítulo 3, a Técnica $T2_{mem}$ explora a identificação de defeitos relativos à memória conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizados as marcações com os estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory», Etapa 1 da Técnica $T2_{mem}$, conforme apresentado na Seção 4.2, Capítulo 4. A Figura 6.5 o diagrama REQ do exemplo APE após as marcações.

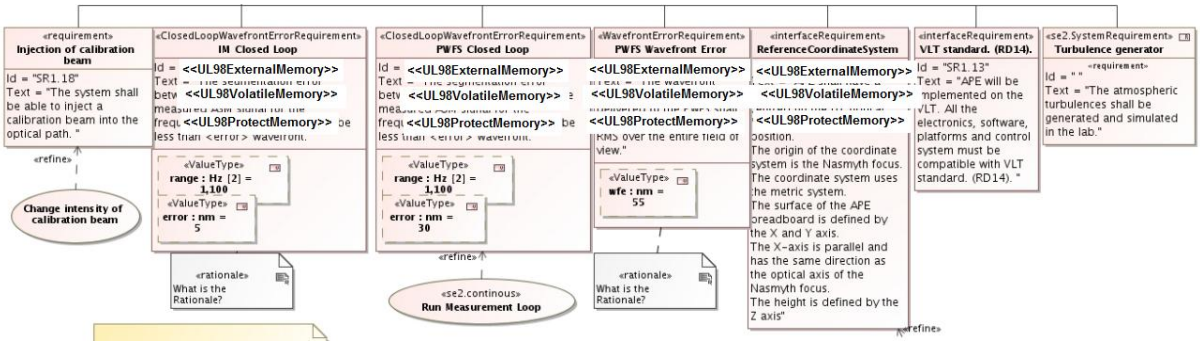


Figura 6.15 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da Técnica $T2_{mem}$.

Etapa II – Preparação do BDD

Na segunda etapa da técnica de leitura, são realizadas as devidas marcações com respeito ao uso dos estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory» no Diagrama BDD, como apresentado nas Diretrizes da Etapa 2 da Técnica $T2_{mem}$, Seção 4.2, Capítulo 4. A Figura 6.16 ilustra os estereótipos marcados no diagrama BDD conforme apresentado anteriormente.

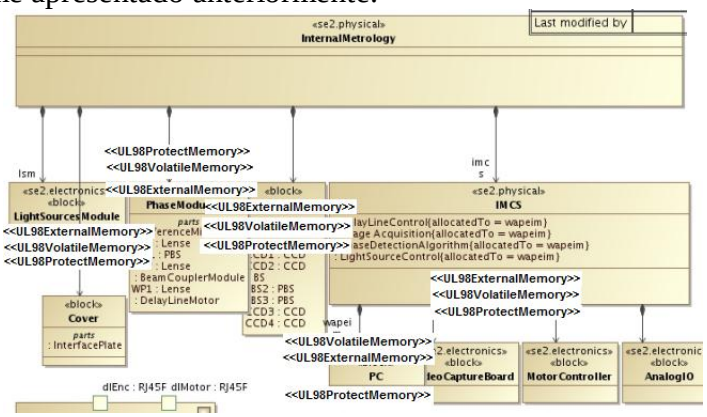


Figura 6.16 – Diagrama BDD (MBSE, 2011) após aplicação da Etapa II da Técnica $T2_{mem}$.

Etapa III – Comparação entre o diagrama REQ com BDD

Após a aplicação das Etapas I e II, e considerando-se o primeiro passo da Etapa III, que pede para verificar se os requisitos marcados no diagrama REQ com os estereótipos mencionados acima estão consistentes com as marcações feitas no BDD, verificou-se que os requisitos "PWFSClosedLoop", "PWFSWavefrontError" e "ReferenceCoordinateSystem" possuíam essas marcações e elas não estavam consistentes com o diagrama BDD. Tais inconsistências foram identificadas a partir das Diretrizes A.1, B.1, C.1, D.1, E.1 e F.1 da Etapa III fazendo com que esses requisitos fossem marcados no diagrama REQ com o estereótipo «IEEEDesignMissing». De modo análogo, para comparar as marcações do diagrama BDD com as marcações do diagrama REQ, pode-se verificar que o havia blocos que não possuía um requisito correspondente. Dessa forma, tais blocos do diagrama BDD foram marcados com o estereótipo «IEEERequirementMissing».

Além disso, à medida que os defeitos foram identificados, o formulário de discrepância foi preenchido conforme as Diretrizes A.2, B.2, C.2, D.2, E.2 e F.2 da técnica $T2_{mem}$. Assim, a Tabela 6.8 apresenta os defeitos detectados e reportados nesse formulário.

Tabela 6.8 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{mem}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Memória**	PWFSClosedLoop	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Memória**	PWFSWaveFrontError	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Memória**	ReferenceCoordinateSystem	«IEEECriticalSeverity»
4	**BDD**	«IEEERequirement Missing»	**Memória**	LightSourceModule	«IEEECriticalSeverity»
5	**BDD**	«IEEERequirement Missing»	**Memória**	PhaseModulator	«IEEECriticalSeverity»
6	**BDD**	«IEEERequirement Missing»	**Memória**	SignalDecoder	«IEEECriticalSeverity»
7	**BDD**	«IEEERequirement Missing»	**Memória**	VideoCaptureBoard	«IEEECriticalSeverity»

Em suma, a partir da Tabela 6.8 pode-se observar que foram detectados 7 defeitos que correspondem ao conceito de memória. Como consequência disso a especificação de requisitos no diagrama REQ e a modelagem no diagrama BDDtornaram-se inconsistentes em relação ao conceito de memória da norma de certificação UL-98. Assim, os defeitos foram classificados como sendo «IEEEDesignMissing» e «IEEERequirementMissing», ambos com alta severidade «IEEECriticalSeverity» uma vez que tais requisitos são essenciais para o funcionamento do sistema APE.

i) Técnica de Leitura $T2_{in}$ (REQ, BDD)

Como exposto no Capítulo 3, a técnica $T2_{in}$ explora a identificação de defeitos relativos a entrada de dados conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa, são realizadas as marcações com os estereótipos «UL98PlausibilityCheck», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{in}$ conforme apresentado na descrição rigorosa da técnica $T2_{in}$. A Figura 4.9 ilustra os estereótipos anotados no diagrama REQ após a aplicação da técnica.

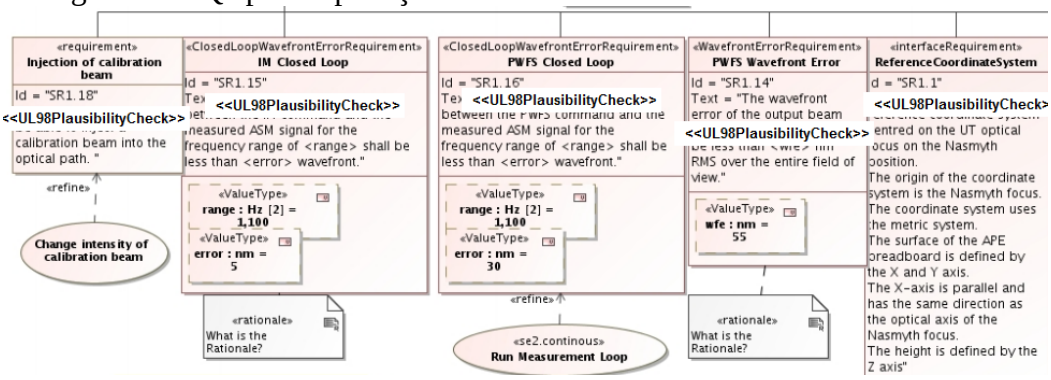


Figura 6.17 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da técnica $T2_{in}$.

Etapa II – Preparação do BDD

Na segunda etapa da técnica, são realizadas as marcações com o estereótipo UL98PlausibilityCheck no diagrama BDD, conforme apresentado na descrição rigorosa da técnica $T2_{in}$. A Figura 6.18 ilustra essas marcações realizadas.

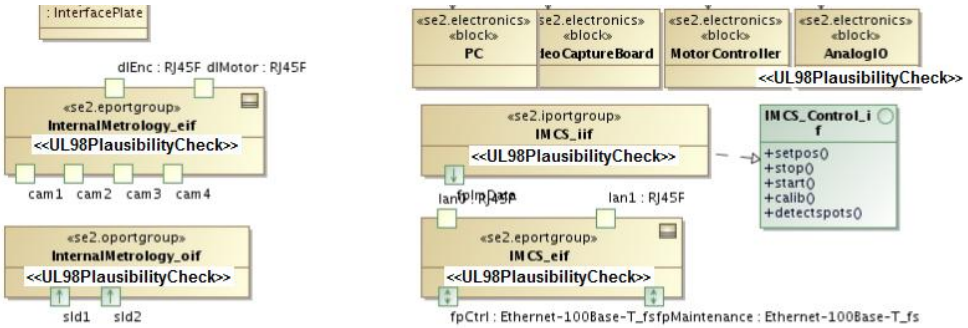


Figura 6.18 – Diagrama BDD (MBSE, 2011) após aplicação da Etapa II da Técnica $T2_{in}$.

Etapa III – Comparação entre o diagrama REQ e BDD

Na terceira etapa da técnica de leitura, são realizadas as verificações comparando-se os Diagramas REQ e BDD simultaneamente com respeito ao estereótipo «UL98PlausibilityCheck» previamente anotados. Assim, a seguinte Tabela 6.9 reportando os defeitos detectados foi elaborada.

Tabela 6.9 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{in}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Entrada**	InjectionOfCalibration	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Entrada**	IMClosedLoop	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Entrada**	PWFSClosedLoop	«IEEECriticalSeverity»
4	**REQ**	«IEEEDesignMissing»	**Entrada**	AnalogIO	«IEEECriticalSeverity»
5	**BDD**	«IEEERequirement Missing»	**Entrada**	InternalMetrology_eif	«IEEECriticalSeverity»
6	**BDD**	«IEEERequirement Missing»	**Entrada**	IMCS_eif	«IEEECriticalSeverity»
7	**BDD**	«IEEESyntax Missing»	**Entrada**	IMCS_iif	«IEEECriticalSeverity»

Após a aplicação da técnica $T2_{in}$, foram observados 7 defeitos dos quais, 4 estão associados com o diagrama REQ e 3 estão associados com o BDD. Desses defeitos associados com o BDD, foi detectado um defeito (defeito 7) corresponde a um defeito sintático no diagrama BDD. Esse defeito, é o resultado da aplicação do passo C, C.1 da Etapa 3 da técnica $T2_{in}$. Nesse caso, tal defeito foi classificado como sendo de severidade média, pois o modelo BDD em relação à esse defeito é parcialmente correto.

j) Técnica de Leitura $T2_{out}$ (REQ, BDD)

Como exposto no Capítulo 3, a técnica $T2_{out}$ explora a identificação de defeitos relativos a saída de dados conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa, são realizadas as marcações com os estereótipos «UL98PlausibilityCheck», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{out}$. A Figura 4.9 ilustra o estereótipo anotado no diagrama REQ após a aplicação da técnica.

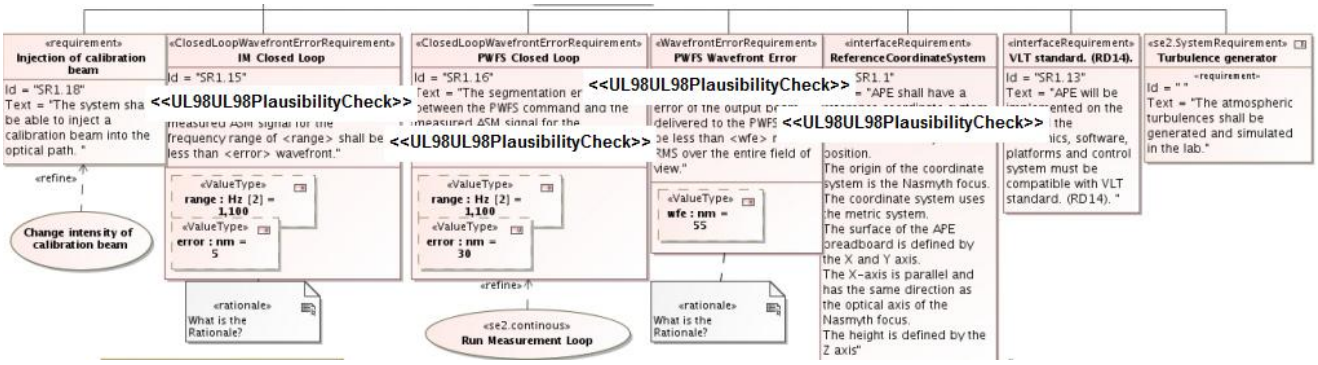


Figura 6.19 - Diagrama REQ (MBSE, 2011) após aplicação da Etapa I da Técnica $T2_{out}$

Etapa II – Preparação do BDD

Na segunda etapa da técnica $T2_{out}$ as seguintes marcações foram realizadas (Figura 6.20).

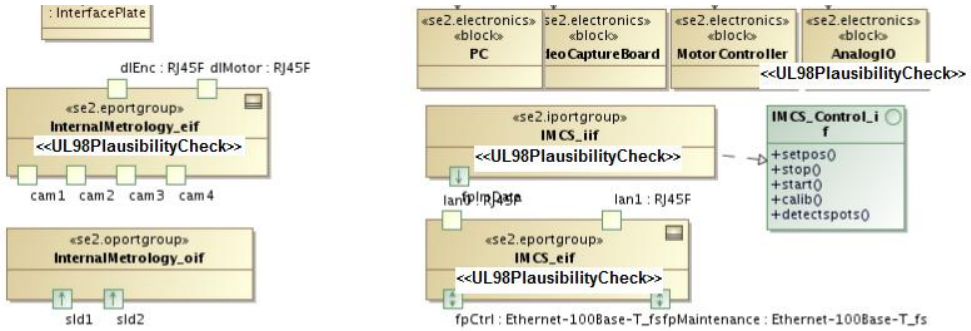


Figura 6.20 – Diagrama BDD (MBSE, 2011) após aplicação da Etapa II da Técnica $T2_{out}$.

Etapa III – Comparação entre o diagrama REQ e BDD

Na terceira etapa da técnica de leitura, são realizadas as verificações comparando-se os Diagramas REQ e BDD simultaneamente com respeito ao estereótipo «UL98PlausibilityCheck» previamente anotado. Assim, a Tabela 6.10 retrata parte do formulário de discrepância com os defeitos detectados.

Após a aplicação da técnica $T2_{out}$, foram detectados 5 defeitos dos quais, 3 estão associados com o diagrama REQ e 2 estão associados com o BDD. Desses defeitos associados

com o BDD, um deles (defeito 5) corresponde a um defeito sintático encontrado no diagrama BDD. Esse defeito, é o resultado da aplicação do passo C, C.1 da Etapa 3 da técnica $T2_{out}$. Nesse caso, tal defeito foi classificado como sendo de severidade média, pois o modelo BDD em relação à esse defeito é parcialmente correto. Note que o bloco anotado com defeito sintático é mesmo do encontrado pela técnica $T2_{in}$, sugerindo que o modelo BDD possui problemas sintáticos em ambos os conceitos tratados pelas técnicas: de entrada e de saída de dados.

Tabela 6.10 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{out}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Saída**	IMClosedLoop	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Saída**	PWFSClosedLoop	«IEEECriticalSeverity»
3	**REQ**	«IEEEDesignMissing»	**Saída**	ReferenceCoordinateSystem	«IEEECriticalSeverity»
4	**BDD**	«IEEERequirement Missing»	**Saída**	PWFSWavefrontError	«IEEECriticalSeverity»
5	**BDD**	«IEEESyntax Missing»	**Saída**	IMCS_iif	«IEEECriticalSeverity»

Como pode se observar, a partir dos resultados encontrados a partir da aplicação da técnica T2 e T1, pode-se verificar que os defeitos detectados anteriormente pela Técnica T1, e que não foram corrigidos, acabaram se propagando da fase de especificação de requisitos do processo SYSMOD para a fase subsequente que á de conhecimento de domínio. Em geral, foi possível observar defeitos relativos a cada um dos conceitos da norma UL-98 (registradores, interrupção, memória e de entrada/saída. Vale salientar, que dentre os defeitos detectados, dois deles chamaram mais a atenção. O primeiro deles diz respeito, ao defeito relativo ao conceito de memória (especificado no bloco de requisito "ReferenceCoordinateSystem") e que não havia um corresponde tanto no IBD quanto no BDD. O segundo deles diz respeito, ao defeito relativo ao conceito de saída, e que está associado ao bloco metrologia "IMClosedLoop", nesse caso, uma restrição operacional de uso de equipamento foi especificada e não foi encontrada tratativa apropriada no diagrama BDD/IBD analisado.

Em resumo, pode ser observado ao longo da aplicação das técnicas RTSS no APE, ocasiões em que defeitos detectados em uma fase do processo SYSMOD foram propagados para fases subsequentes. Por exemplo, logo no início do desenvolvimento dessa aplicação, houve defeitos identificados no diagrama REQ que passaram para o diagrama BDD pelo fato de não serem adequadamente modelados no diagrama IBD. Essa propagação de defeitos, muito provavelmente, gerou retrabalho em algum momento do processo de desenvolvimento, uma vez que muitos desses defeitos caracterizam a ausência de componentes para que a aplicação funcione adequadamente. Assim, nesse exemplo APE, caso as técnicas de leitura RTSS fossem aplicadas à medida que os diagramas eram concluídos, os defeitos poderiam ser corrigidos na própria fase em que eram inseridos, evitando-se a eventual propagação de defeitos e, consequentemente, re-trabalho futuro e aumento no custo de desenvolvimento.

6.3 Computador de Bordo

Este exemplo explora a aplicação das técnicas RTSS para o exemplo disponível no Livro "*Systems Enginnering with SysML/UML: Modeling, Analysis and Design*" (WEILKIENS, 2008). O exemplo do livro descreve um Sistema Embarcado projetado com o processo SYSMOD. Nesse exemplo, o autor descreve todos os artefatos utilizados para modelar o projeto de um sistema de computador de bordo, que é embutido no compartimento DIN de veículos automotivos de uma agência de aluguel fictícia. Esse sistema contempla os requisitos descritos na Tabela 6.11.

Tabela 6.11 - Requisitos do Sistema Embarcado Computador de Bordo (adaptado (WEILKIENS, 2008).

#	Requisito
1	O sistema deve fornecer facilidade e conforto para ter acesso ao veículo de aluguel.
2	O sistema deve fornecer uma interface disponível no computador de bordo para reservar um veículo de aluguel.
3	O sistema deve realizar a reserva de um veículo com o uso de mensagens de SMS.
4	O sistema deve fornecer latência e tempo de resposta adequado para a troca de mensagens via SMS.
5	O sistema deve prover modos para realizar a instalação, configuração e configuração dos serviços disponíveis no computador de bordo.
6	O sistema deve prover meios de acesso no computador de bordo à navegação via GPS.
7	O sistema deve integrar obrigatoriamente o compartimento DIN de Rádio de veículos automotores.

Os requisitos apresentados na Tabela 6.11 estão sendo retratados a seguir no conjunto dos diagramas REQ (Figuras 6.21), IBD (Figura 6.22), BDD (Figura 6.23), SD (Figura 6.24) e SMD (Figura 6.25). Nesse exemplo em particular, optou-se por retratar os diagramas de entrada utilizados pelas técnicas RTSS no início da seção (e não ao longo da aplicação das técnicas) uma vez que a forma de apresentação das marcações segue o mesmo princípio do exporto da anteriormente (Seção 6.2). Além disso, em cada etapa de aplicação das técnicas (Etapa I, Etapa II e Etapa III), foram utilizadas tabelas que caracterizam o relacionamento de cada marcação requerida pela técnica com o respectivo elemento do diagrama inspecionado i.e., bloco, requisito, etc..

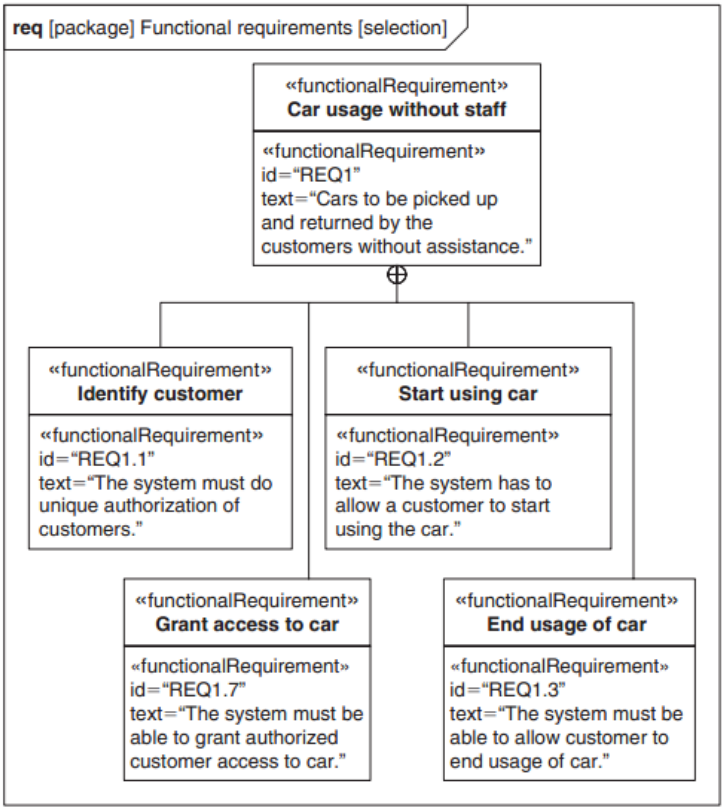


Figura 6.21 - Diagrama de Requisitos SysML/SYSMOD(WEILKIENS, 2008).

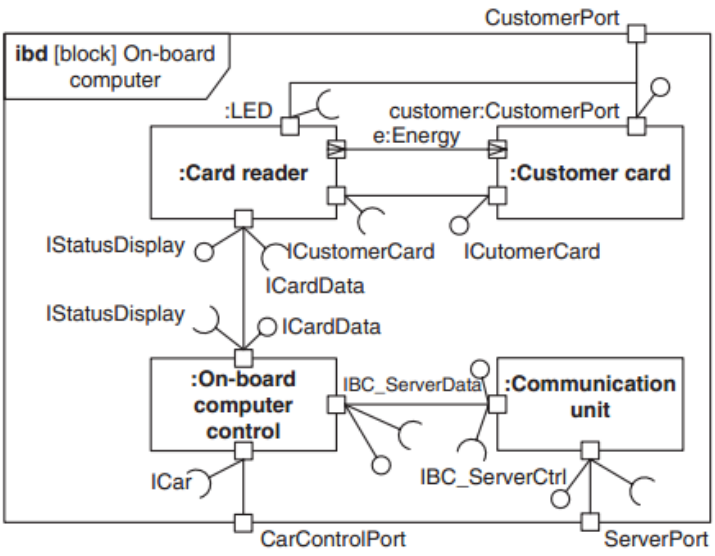


Figura 6.22 - Diagrama Interno de Bloco da SysML/SYSMOD(WEILKIENS, 2008).

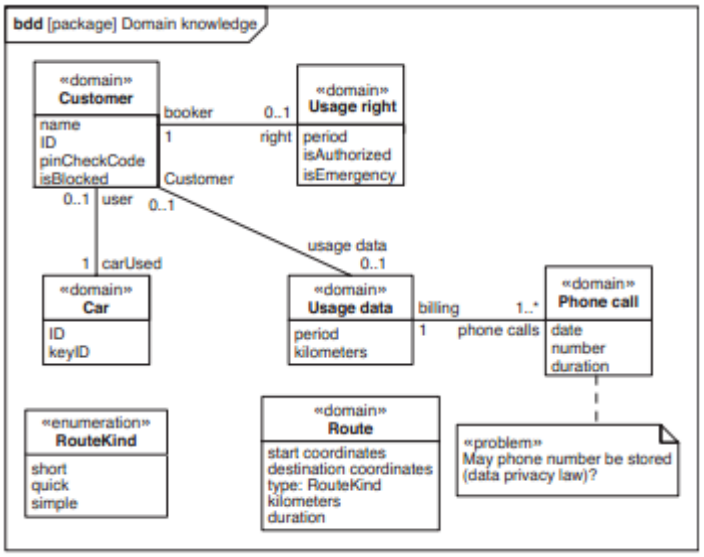


Figura 6.23 - Diagrama de Definição de Bloco da SysML/SYSMOD(WEILKIENS, 2008).

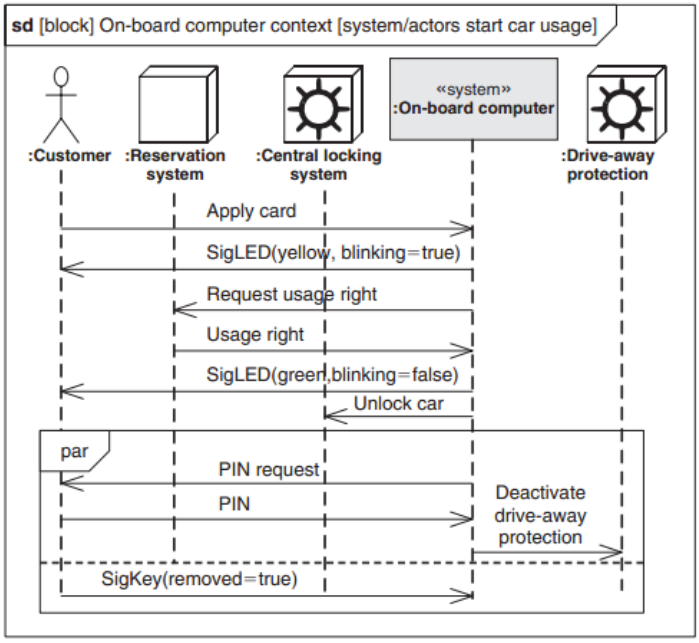


Figura 6.24 - Diagrama de Sequência da SysML/SYSMOD(WEILKIENS, 2008).

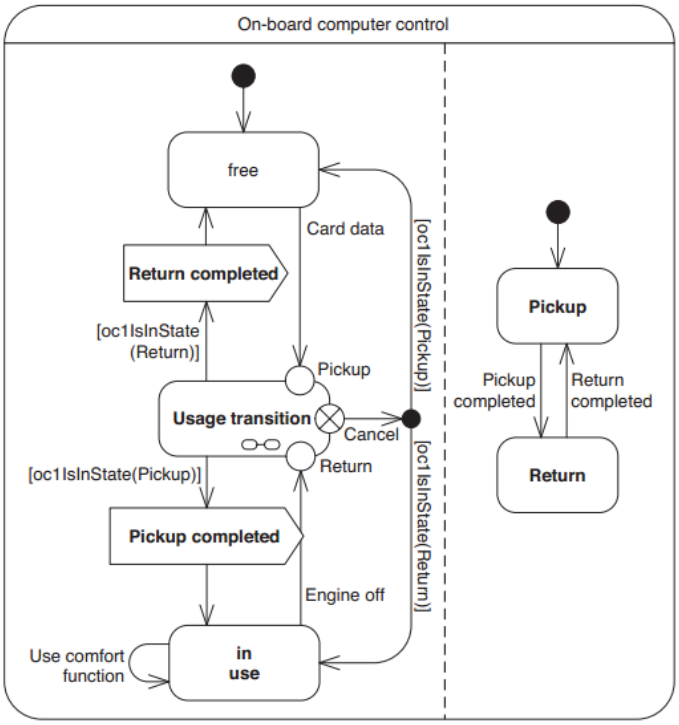


Figura 6.25 - Diagrama de Máquina de Estados da SysML/SYSMOD(WEILKIENS, 2008).

a) Técnica de Leitura $T1_{reg}$ (REQ, IBD)

Como exposto no Capítulo 3, a técnica $T1_{reg}$ explora a identificação de defeitos relativos a registradores conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «UL98CPUProg» e «UL98CPUReg», como indicado nas Diretrizes da Etapa 1 da Técnica $T1_{reg}$. Observando o diagrama REQ (Figura 6.21) foi possível realizar as marcações (✓) mostradas na Tabela 6.12:

Tabela 6.12 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T1_{reg}$.

Requisito	«UL98CPUProg»	«UL98CPUReg»
IdentifyCustomer GrantAccessToCar StartUsingCar EndUsageOfCar	✓	✓

Etapa II – Preparação do IBD

Na segunda etapa da técnica de leitura, são realizadas as devidas marcações com os estereótipos «UL98CPUProg» e «UL98CPUReg» no Diagrama IBD, como apresentado nas Diretrizes da Etapa 2 da Técnica $T1_{reg}$. Observando o diagrama IBD (Figura 6.22) foi possível realizar as marcações mostradas na Tabela 6.13.

Tabela 6.13 - Marcações realizadas no diagrama IBD após aplicação da Etapa II da Técnica $T1_{reg}$.

Bloco	«UL98CPUProg»	«UL98CPUReg»
CardReader		
CustomerCard		
OnBoardComputerControl	✓	✓
CommunicationUnit	✓	✓

Etapa III – Comparação entre o diagrama REQ e IBD

Na terceira etapa da Técnica de Leitura, são realizadas as verificações comparando os Diagramas IBD e REQ simultaneamente com respeito aos estereótipos «UL98CPUProg» e «UL98CPUReg»previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T1_{reg}$. Dessa forma, a Tabela 6.14, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.14 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{reg}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEE DesignMissing»	**Registrador**	GrantAccessToCar	«IEEECriticalSeverity»
2	**REQ**	«IEEE DesignMissing»	**Registrador**	StartUsingCar	«IEEECriticalSeverity»
3	**IBD**	«IEEE ReqMissing»	**Registrador**	OnBoardComputerControl	«IEEECriticalSeverity»
4	**IBD**	«IEEE ReqMissing»	**Registrador**	CommunicationUnit	«IEEECriticalSeverity»

Após a aplicação da Etapa III, pode-se verificar que haviam 4 defeitos dos quais 2 eram relativos ao diagrama REQ e 2 eram relativos ao IBD. Assim, tais defeitos indicam que os diagramas não são consistentes com respeito ao conceito de registrador da norma de certificação UL-98. Um exemplo importante dessa inconsistência, indicado pelo primeiro defeito, é a ausência de uma especificação no diagrama IBD relativa à ao acesso do sistema de computador de bordo "GrantAccessToCar". Nesse caso, considerando-se que a garantia de acesso a pessoas autorizadas é um requisito funcional do sistema de computador de bordo, o mesmo deveria ter sido adequadamente representado na forma de um bloco no diagrama IBD.

b) Técnica de Leitura $T1_{int}$ (REQ, IBD)

Como exposto no Capítulo 3, a técnica $T1_{int}$ explora a identificação de defeitos relativos a interrupções conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizados as marcações com auxílio dos estereótipos «UL98Interrupt» e «UL98Clock», comoindicado nas Diretrizes da Etapa 1 da Técnica $T1_{int}$, conforme apresentado na descrição rigorosa da técnica $T1_{int}$. Observando o diagrama REQ (Figura 6.21) foi possível realizar as marcações (✓) mostradas na Tabela 6.15:

Tabela 6.15 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da técnica $T1_{int}$.

Requisito	«UL98Interrupt»	«UL98Clock»
IdentifyCustomer	✓	
GrantAccessToCar	✓	
StartUsingCar	✓	
EndUsageOfCar	✓	

Etapa II – Preparação do IBD

Na segunda etapa da Técnica de Leitura, são realizados as devidas marcações com respeito ao uso dos estereótipos «UL98Interrupt» e «UL98Clock» no Diagrama BDD. Observando o diagrama IBD (Figura 6.22) foi possível realizar as marcações (✓) mostradas na Tabela 6.16:

Tabela 6.16 - Marcações realizadas no diagrama IBD após aplicação da Etapa II da técnica $T1_{int}$.

Bloco	«UL98Interrupt»	«UL98Clock»
CardReader	✓	
CustomerCard	✓	
OnBoardComputerControl	✓	
CommunicationUnit	✓	

Etapa III – Comparação entre o diagrama REQ e IBD

Na terceira etapa da Técnica de Leitura, são realizados as verificações comparando os Diagramas IBD e REQ simultaneamente com respeito aos estereótipos «UL98Interrupt» e «UL98Clock», previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T1_{int}$. Dessa forma, a Tabela 6.17, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.17 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{int}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Interrupção**	StartUsingCar	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Interrupção**	EndUsageOfCar	«IEEECriticalSeverity»
3	**IBD**	«IEEEReqMissing»	**Interrupção**	OnBoardComputerControl	«IEEECriticalSeverity»
4	**IBD**	«IEEEReqMissing»	**Interrupção**	CommunicationUnit	«IEEECriticalSeverity»

Após a aplicação da Etapa III, pode-se verificar que foram detectados 4 defeitos dos quais, 2 eram relativos ao diagrama REQ e 2 eram relativos ao IBD. Assim, tais defeitos indicam que os diagramas não são consistentes e como consequência, informações importantes não foram encontradas (tanto no REQ quanto no IBD). Como exemplo, o defeito 3 (vide Tabela 6.17), está associado com o bloco "OnBoardControl" que não possui especificação adequada no REQ, como consequência, tal bloco pode ser mal implementado levando a uma falha do sistema de computador de bordo.

c) Técnica de Leitura $T1_{mem}$ (REQ, IBD)

Como exposto no Capítulo 3, a técnica $T1_{mem}$ explora a identificação de defeitos com memória conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory», como indicado nas Diretrizes da Etapa 1 da Técnica $T1_{mem}$. Observando o diagrama REQ (Figura 6.21) foi possível realizar as marcações (✓) mostradas na Tabela 6.18.

Tabela 6.18 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da técnica $T1_{mem}$.

Requisito	«UL98ExternalMemory»	«UL98VolatileMemory»	«UL98ProtectMemory»
IdentifyCustomer	✓	✓	✓
StartUsingCar	✓	✓	✓
GrandAccessToCar	✓	✓	✓
EndUsageOfCar	✓	✓	✓

Etapa II – Preparação do IBD

De modo análogo à Etapa I, nessa etapa foram feitas as marcações com osestereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory» no diagrama IBD. A Tabela 6.19, mostra as marcações (✓) realizadas no diagrama IBD (Figura 6.22).

Tabela 6.19 - Marcações realizadas no diagrama IBD após aplicação da Etapa II da técnica $T1_{mem}$.

Bloco	«UL98ExternalMemory»	«UL98VolatileMemory»	«UL98ProtectMemory»
CardReader	✓	✓	✓
CustomerCard	✓	✓	✓
OnBoardComputerControl	✓	✓	✓
ConnunicationUnit	✓	✓	✓

Etapa III – Comparação entre o REQ e IBD

Na terceira etapa da Técnica de Leitura, são realizadas as verificações comparando os Diagramas IBD e REQ simultaneamente com respeito aos estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory» previamente anotados da Etapa 3 da técnica $T1_{mem}$. Dessa forma, a Tabela 6.20, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.20 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{mem}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEE DesignMissing»	**Memória**	StartUsingCar	«IEEECriticalSeverity»
2	**REQ**	«IEEE DesignMissing»	**Memória**	EndUsageOfCar	«IEEECriticalSeverity»
3	**IBD**	«IEEE ReqMissing»	**Memória**	OnBoardComputerControl	«IEEECriticalSeverity»
4	**IBD**	«IEEE ReqMissing»	**Memória**	ConnunicationUnit	«IEEECriticalSeverity»

d) Técnica de Leitura $T1_{in}$ (REQ, IBD)

Como exposto no Capítulo 3, a técnica $T1_{in}$ explora a identificação de defeitos relativos a entrada de dados conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa, são realizadas as marcações com os estereótipos «UL98PlausibilityCheck», como indicado nas Diretrizes da Etapa 1 da Técnica $T1_{in}$ conforme apresentado na descrição rigorosa da técnica $T1_{in}$. A Tabela 6.21, mostra as marcações (✓) realizadas no diagrama IBD (Figura 6.22).

Tabela 6.21 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T1_{in}$.

Requisito	«UL98PlausibilityCheck»
IdentifyCustomer	✓
StartUsingCar	✓
GrandAccessToCar	✓
EndUsageOfCar	✓

Etapa II – Preparação do IBD

De modo análogo à Etapa I, foram feitas as marcações com osestereótipos«UL98PlausibilityCheck» no diagrama IBD. A Tabela 6.22, mostra as marcações (✓) realizadas no diagrama IBD (Figura 6.22).

Tabela 6.22 - Marcações realizadas no diagrama IBD após aplicação da Etapa II da Técnica $T1_{out}$.

Bloco	«UL98PlausibilityCheck»
CardReader	✓
CustomerCard	✓
OnBoardComputerControl	✓
ConnunicationUnit	✓

Etapa III – Comparação entre REQ e IBD

Na terceira etapa da Técnica de Leitura, são realizados as verificações comparando os Diagramas IBD e REQ simultaneamente com respeito aos estereótipos«UL98PlausibilityCheck» previamente anotados, comoindicado nas Diretrizes da Etapa 3 da técnica $T1_{in}$. Dessa forma, a Tabela 6.23, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.23 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{in}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Entrada**	StartUsingCar	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Entrada**	EndUsageOfCar	«IEEECriticalSeverity»
3	**IBD**	«IEEEReqMissing»	**Entrada**	OnBoardComputerControl	«IEEECriticalSeverity»
4	**IBD**	«IEEEReqMissing»	**Entrada**	ConnunicationUnit	«IEEECriticalSeverity»

e) Técnica de Leitura $T1_{out}$(REQ, IBD)

Como exposto no Capítulo 3, a técnica $T1_{in}$ explora a identificação de defeitos relativos a saída de dados conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e IBD.

Etapa I – Preparação do REQ

Na primeira etapa, são realizadas as marcações com os estereótipos «UL98PlausibilityCheck», como indicado nas Diretrizes da Etapa 1 da Técnica $T1_{out}$ conforme apresentado na descrição rigorosa da Técnica $T1_{out}$. A Tabela 6.24, mostra as marcações (✓) realizadas no diagrama REQ (Figura 6.21).

Tabela 6.24 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T1_{out}$.

Requisito	«UL98PlausibilityCheck»
IdentifyCustomer	✓
StartUsingCar	✓
GrandAccessToCar	✓
EndUsageOfCar	✓

Etapa II – Preparação do IBD

De modo análogo à Etapa I, foram feitas as marcações com os estereótipos «UL98PlausibilityCheck» no diagrama IBD. A Tabela 6.25, mostra as marcações (✓) realizadas no diagrama IBD (Figura 6.22).

Tabela 6.25 - Marcações realizadas no diagrama IBD após aplicação da Etapa II da Técnica $T1_{out}$.

Bloco	«UL98PlausibilityCheck»
CardReader	✓
CustomerCard	✓
OnBoardComputerControl	✓
ConnunicationUnit	✓

Etapa III – Comparação entre REQ e IBD

Na terceira etapa da Técnica de Leitura, são realizados as verificações comparando os Diagramas IBD e REQ simultaneamente com respeito aos estereótipos «UL98PlausibilityCheck» previamente anotados, comoindicado nas Diretrizes da Etapa 3 da Técnica $T1_{out}$. Dessa forma, a Tabela 6.26, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.26 - Parte do Formulário de Discrepância após aplicação da Técnica $T1_{out}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEE DesignMissing»	**Saída**	StartUsingCar	«IEEECriticalSeverity»
2	**REQ**	«IEEE DesignMissing»	**Saída**	EndUsageOfCar	«IEEECriticalSeverity»
3	**IBD**	«IEEE ReqMissing»	**Saída**	OnBoardComputerControl	«IEEECriticalSeverity»
4	**IBD**	«IEEE ReqMissing»	**Saída**	ConnunicationUnit	«IEEECriticalSeverity»

f) Técnica de Leitura $T2_{reg}$ (REQ, BDD)

Como exposto no Capítulo 3, a técnica $T2_{reg}$ explora a identificação de defeitos relativos a registradores conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «UL98CPUProg» e «UL98CPUReg», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{reg}$. Observando o diagrama REQ (Figura 6.21) foi possível realizar as marcações (✓) mostradas na Tabela 6.27:

Tabela 6.27 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T2_{reg}$.

Requisito	«UL98CPUProg»	«UL98CPUReg»
IdentifyCustomer		
GrantAccessToCar		✓
StartUsingCar	✓	
EndUsageOfCar		

Etapa II – Preparação do BDD

Na segunda etapa da técnica de leitura, são realizadas as devidas marcações com os estereótipos «UL98CPUProg» e «UL98CPUReg» no Diagrama IBD, comoapresentado nas Diretrizes da Etapa 2 da Técnica $T2_{reg}$. Observando o diagrama BDD (Figura 6.23) foi possível realizar as marcações mostradas na Tabela 6.28.

Tabela 6.28 - Marcações realizadas no diagrama BDD após aplicação da Etapa II da Técnica $T2_{reg}$.

Bloco	«UL98CPUProg»	«UL98CPUReg»
Customer	✓	✓
Usage Right		✓
Route		✓

Etapa III – Comparação entre o diagrama REQ eBDD

Na terceira etapa da Técnica de Leitura, são realizados as verificações comparando os Diagramas BDD e REQ simultaneamente com respeito aos estereótipos «UL98CPUProg» e «UL98CPUReg»previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T2_{reg}$. Dessa forma, a Tabela 6.29, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.29 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{reg}$.

Def.	Diagrama	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	IEEE DesignMissing	**Registrador**	StartUsingCar	«IEEECritical Severity»
2	**BDD**	IEEE ReqMissing	**Registrador**	Customer	«IEEECritical Severity»

g) Técnica de Leitura $T2_{int}$ (REQ, BDD)

Como exposto no Capítulo 3, a técnica $T2_{int}$ explora a identificação de defeitos relativos a interrupções conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizados as marcações com auxílio dos estereótipos «UL98Interrupt» e «UL98Clock», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{int}$, conforme apresentado na descrição rigorosa da técnica $T2_{int}$. Observando o diagrama REQ (Figura 6.21) foi possível realizar as marcações (✓) mostradas na Tabela 6.30:

Tabela 6.30 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T2_{int}$.

Requisito	«UL98Interrupt»	«UL98Clock»
IdentifyCustomer	✓	
GrantAccessToCar	✓	
StartUsingCar	✓	
EndUsageOfCar	✓	

Etapa II – Preparação do BDD

Na segunda etapa da Técnica de Leitura, são realizados as devidas marcações com respeito ao uso dos estereótipos «UL98Interrupt» e «UL98Clock» no Diagrama BDD. Observando o diagrama BDD (Figura 6.23) foi possível realizar as marcações (✓) mostradas na Tabela 6.31:

Tabela 6.31 - Marcações realizadas no diagrama BDD após aplicação da Etapa II da Técnica $T2_{int}$.

Bloco	«UL98Interrupt»	«UL98Clock»
UsageRight	✓	
UsageData		✓
Route		✓

Etapa III – Comparação entre o diagrama REQ e BDD

Na terceira etapa da Técnica de Leitura, são realizados as verificações comparando os Diagramas BDD e REQ simultaneamente com respeito aos estereótipos «UL98Interrupt» e «UL98Clock» previamente anotados, como indicado nas Diretrizes da Etapa 3 da Técnica $T2_{int}$. Dessa forma, a Tabela 6.32, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.32 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{int}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**BDD**	«IEEEReqMissing»	**Interrupção**	UsageData	«IEEECriticalSeverity»
2	**BDD**	«IEEEReqMissing»	**Interrupção**	Route	«IEEECriticalSeverity»

h) Técnica de Leitura $T2_{mem}$ (REQ, BDD)

Como exposto no Capítulo 3, a técnica $T2_{mem}$ explora a identificação de defeitos relativos a memória conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{mem}$. Observando o diagrama REQ (Figura 6.21) foi possível realizar as marcações (✓) mostradas na Tabela 6.33.

Tabela 6.33 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T2_{mem}$.

Requisito	«UL98ExternalMemory»	«UL98VolatileMemory»	«UL98ProtectMemory»
IdentifyCustomer	✓	✓	✓
StartUsingCar	✓	✓	✓
GrandAccessToCar	✓	✓	✓
EndUsageOfCar	✓	✓	✓

Etapa II – Preparação do BDD

De modo análogo à Etapa I, nessa etapa foram feitas as marcações com os estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory» no diagrama IBD. A Tabela 6.34, mostra as marcações (✓) realizadas no diagrama BDD (Figura 6.23).

Tabela 6.34 - Marcações realizadas no diagrama BDD após aplicação da Etapa II da Técnica $T2_{mem}$.

Bloco	«UL98ExternalMemory»	«UL98VolatileMemory»	«UL98ProtectMemory»
Customer	✓	✓	✓
Car	✓	✓	✓
UsageRight	✓	✓	✓
UsageData	✓	✓	✓
PhoneCall	✓	✓	✓
Route	✓	✓	✓
RouteKind	✓	✓	✓

Etapa III – Comparação entre o REQ eBDD

Na terceira etapa da Técnica de Leitura, são realizados as verificações comparando os Diagramas BDD e REQ simultaneamente com respeito aos estereótipos «UL98ExternalMemory», «UL98VolatileMemory» e «UL98ProtectMemory» previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T2_{mem}$. Dessa forma, a Tabela 6.35, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.35 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{mem}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Memória**	StartUsingCar	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Memória**	EndUsageOfCar	«IEEECriticalSeverity»
3	**BDD**	«IEEEReqMissing»	**Memória**	UsageData	«IEEECriticalSeverity»
4	**BDD**	«IEEEReqMissing»	**Memória**	PhoneCall	«IEEECriticalSeverity»
5	**BDD**	«IEEEReqMissing»	**Memória**	Route	«IEEECriticalSeverity»
6	**BDD**	«IEEEReqMissing»	**Memória**	RouteKind	«IEEECriticalSeverity»

i) Técnica de Leitura $T2_{in}$ (REQ, BDD)

Como exposto no Capítulo 3, a Técnica $T2_{in}$ explora a identificação de defeitos relativos a entrada de dados conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa, são realizadas as marcações com os estereótipos «UL98PlausibilityCheck», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{in}$ conforme apresentado na descrição rigorosa da técnica $T2_{in}$. A Tabela 6.36, mostra as marcações (✓) realizadas no diagrama BDD (Figura 6.21).

Tabela 6.36 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T2_{in}$.

Requisito	«UL98PlausibilityCheck»
IdentifyCustomer	✓
StartUsingCar	✓
GrandAccessToCar	✓
EndUsageOfCar	✓

Etapa II – Preparação do BDD

De modo análogo à Etapa I, foram feitas as marcações com os estereótipos «UL98PlausibilityCheck» no diagrama BDD. A Tabela 6.37, mostra as marcações (✓) realizadas no diagrama BDD (Figura 6.23).

Tabela 6.37 - Marcações realizadas no diagrama BDD após aplicação da Etapa II da Técnica $T2_{in}$.

Bloco	«UL98PlausibilityCheck»
Customer	✓
Car	✓
UsageRight	✓
UsageData	✓
PhoneCall	✓
Route	✓
RouteKind	✓

Etapa III – Comparação entre REQ e BDD

Na terceira etapa da Técnica de Leitura, são realizados as verificações comparando os Diagramas BDD e REQ simultaneamente com respeito aos estereótipos «UL98PlausibilityCheck» previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T2_{in}$. Dessa forma, a Tabela 6.38, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.38 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{in}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Entrada**	StartUsingCar	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Entrada**	EndUsageOfCar	«IEEECriticalSeverity»
3	**BDD**	«IEEEReqMissing»	**Entrada**	PhoneCall	«IEEECriticalSeverity»
4	**BDD**	«IEEEReqMissing»	**Entrada**	Route	«IEEECriticalSeverity»
5	**BDD**	«IEEEReqMissing»	**Entrada**	RouteKind	«IEEECriticalSeverity»

j) Técnica de Leitura $T2_{out}$(REQ, BDD)

Como exposto no Capítulo 3, a técnica $T2_{out}$ explora a identificação de defeitos relativos a saída de dados conforme a cobertura requerida pela UL-98. Os artefatos de entrada para essa técnica são os diagramas: REQ e BDD.

Etapa I – Preparação do REQ

Na primeira etapa, são realizadas as marcações com os estereótipos «UL98PlausibilityCheck», como indicado nas Diretrizes da Etapa 1 da Técnica $T2_{out}$ conforme apresentado na descrição rigorosa da técnica $T2_{out}$. A Tabela 6.39, mostra as marcações (✓) realizadas no diagrama BDD (Figura 6.23).

Tabela 6.39 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T2_{out}$.

Requisito	«UL98PlausibilityCheck»
IdentifyCustomer	✓
StartUsingCar	✓
GrandAccessToCar	✓
EndUsageOfCar	✓

Etapa II – Preparação do IBD

De modo análogo à Etapa I, foram feitas as marcações com os estereótipos «UL98PlausibilityCheck» no diagrama BDD. A Tabela 6.40, mostra as marcações (✓) realizadas no diagrama BDD (Figura 6.23).

Tabela 6.40 - Marcações realizadas no diagrama IBD após aplicação da Etapa II da Técnica $T2_{out.}$

Bloco	«UL98PlausibilityCheck»
Customer	✓
Car	✓
UsageRight	✓
UsageData	✓
PhoneCall	✓
Route	✓
RouteKind	✓

Etapa III – Comparação entre REQ eBDD

Na terceira etapa da Técnica de Leitura, são realizadas as verificações comparando os Diagramas BDD e REQ simultaneamente com respeito aos estereótipos «UL98PlausibilityCheck» previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T2_{out}$. Dessa forma, a Tabela 6.41, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.41 - Parte do Formulário de Discrepância após aplicação da Técnica $T2_{out}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**REQ**	«IEEEDesignMissing»	**Saída**	StartUsingCar	«IEEECriticalSeverity»
2	**REQ**	«IEEEDesignMissing»	**Saída**	EndUsageOfCar	«IEEECriticalSeverity»
3	**BDD**	«IEEEReqMissing»	**Saída**	Route	«IEEECriticalSeverity»
4	**BDD**	«IEEEReqMissing»	**Saída**	RouteKind	«IEEECriticalSeverity»

k) Técnica de Leitura $T3_{par}$ (SEQ, REQ)

Como exposto no Capítulo 3, a técnica $T3_{par}$ explora a identificação de defeitos relativos aos elementos de paralelismo (concorrência) da SysML. Os artefatos de entrada para essa técnica são os diagramas: SEQ e REQ.

Etapa I – Preparação do SEQ

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «Parallel» e «NonParallel», como indicado nas Diretrizes da Etapa 1 da Técnica $T3_{par}$. Observando o diagrama SEQ (Figura 6.24) foi possível realizar as marcações (✓) mostradas na Tabela 6.42:

Tabela 6.42 - Marcações realizadas no diagrama SEQ após aplicação da Etapa I da Técnica $T3_{par}$.

Blocos	«Parallel»	«NonParallel»
Customer	✓	
ReservationSystem	✓	
CentralSystem	✓	
OnBoardComputer	✓	
DriveAwayProtection	✓	

Etapa II – Preparação do REQ

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «ReqParallel», como indicado nas Diretrizes da Etapa 2 da Técnica $T3_{par}$. Observando o diagrama REQ (Figura 6.21) não foi possível identificar marcações no diagrama REQ.

Etapa III – Comparação entre SEQ e REQ

Na terceira etapa da Técnica de Leitura, são realizados as verificações comparando os Diagramas SEQ e REQ simultaneamente com respeito aos estereótipos «Parallel»,«NonParallel» e «ReqParallel» previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T3_{par}$. Dessa forma, a Tabela 6.43, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.43 - Parte do Formulário de Discrepância após aplicação da Técnica $T3_{par}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito	Severidade
1	**SEQ**	«IEEEReqMissing»	**Paralelismo**	Customer	«IEEECriticalSeverity»
2	**SEQ**	«IEEEReqMissing»	**Paralelismo**	ReservationSystem	«IEEECriticalSeverity»
3	**SEQ**	«IEEEReqMissing»	**Paralelismo**	CentralSystem	«IEEECriticalSeverity»
4	**SEQ**	«IEEEReqMissing»	**Paralelismo**	OnBoardComputer	«IEEECriticalSeverity»
5	**SEQ**	«IEEEReqMissing»	**Paralelismo**	DriveAwayProtection	«IEEECriticalSeverity»

l) Técnica de Leitura $T4_{comp}$ (REQ, MEF)

Como exposto no Capítulo 3, a Técnica $T4_{comp}$ explora a identificação de defeitos relativos a hierarquia. Os artefatos de entrada para essa técnica são os diagramas: REQ e MEF.

Etapa I – Preparação do REQ

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «IEEESyntaxMissing», «EssReq» e «TecReq» como indicado nas Diretrizes da Etapa 1 da Técnica $T4_{comp}$. Observando o diagrama REQ (Figura 6.21) foi possível realizar as marcações (✓) mostradas na Tabela 6.44:

Tabela 6.44 - Marcações realizadas no diagrama REQ após aplicação da Etapa I da Técnica $T4_{comp}$.

Requisito	«IEEESyntaxMissing»	«EssReq»	«TecReq»
CarUsageWithoutStaff		✓	
IdentifyCustomer			✓
StartUsingCar			✓
GrandAccessToCar			✓
EndUsageOfCar			✓

Etapa II – Preparação da MEF

Análogo a etapa anterior as seguintes marcações (✓) foram feitas no diagrama MEF como mostrada a Tabela 6.45:

Tabela 6.45 - Marcações realizadas no diagrama REQ após aplicação da Etapa II da Técnica $T4_{comp}$.

Estados	«IEEESyntaxMissing»	«SuperState»	«SubState»
OnBoard Control		✓	
Free			✓
UsageTransition		✓	
InUse			✓
Pickup			✓
Return			✓

Etapa III – Comparação do REQ com MEF

Na terceira etapa da técnica, são realizados as verificações comparando os diagramas MEF e REQ simultaneamente com respeito aos estereótipos «SuperState», «SubState», «EssReq» e «TecRec» previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T4_{comp}$. Dessa forma, é possível identificar os seguintes requisitos e/ou blocos internos com defeitos como apresentado na Tabela 6.46.

Tabela 6.46 - Parte do Formulário de Discrepância após aplicação da técnica $T4_{comp}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	**MEF**	«IEEEReqMissing»	**Hierarquia**	UsageTransition	«IEEECritical Severity»
2	**MEF**	«IEEEReqMissing»	**Hierarquia**	InUse	«IEEECritical Severity»
3	**MEF**	«IEEEReqMissing»	**Hierarquia**	Pickup	«IEEECritical Severity»
4	**MEF**	«IEEEReqMissing»	**Hierarquia**	Return	«IEEECritical Severity»

Em suma, para sintetizar a existência de propagação de defeitos entre as fases de desenvolvimento do processo SYSMOD foi elaborada a Figura 6.26 ilustrada a seguir.

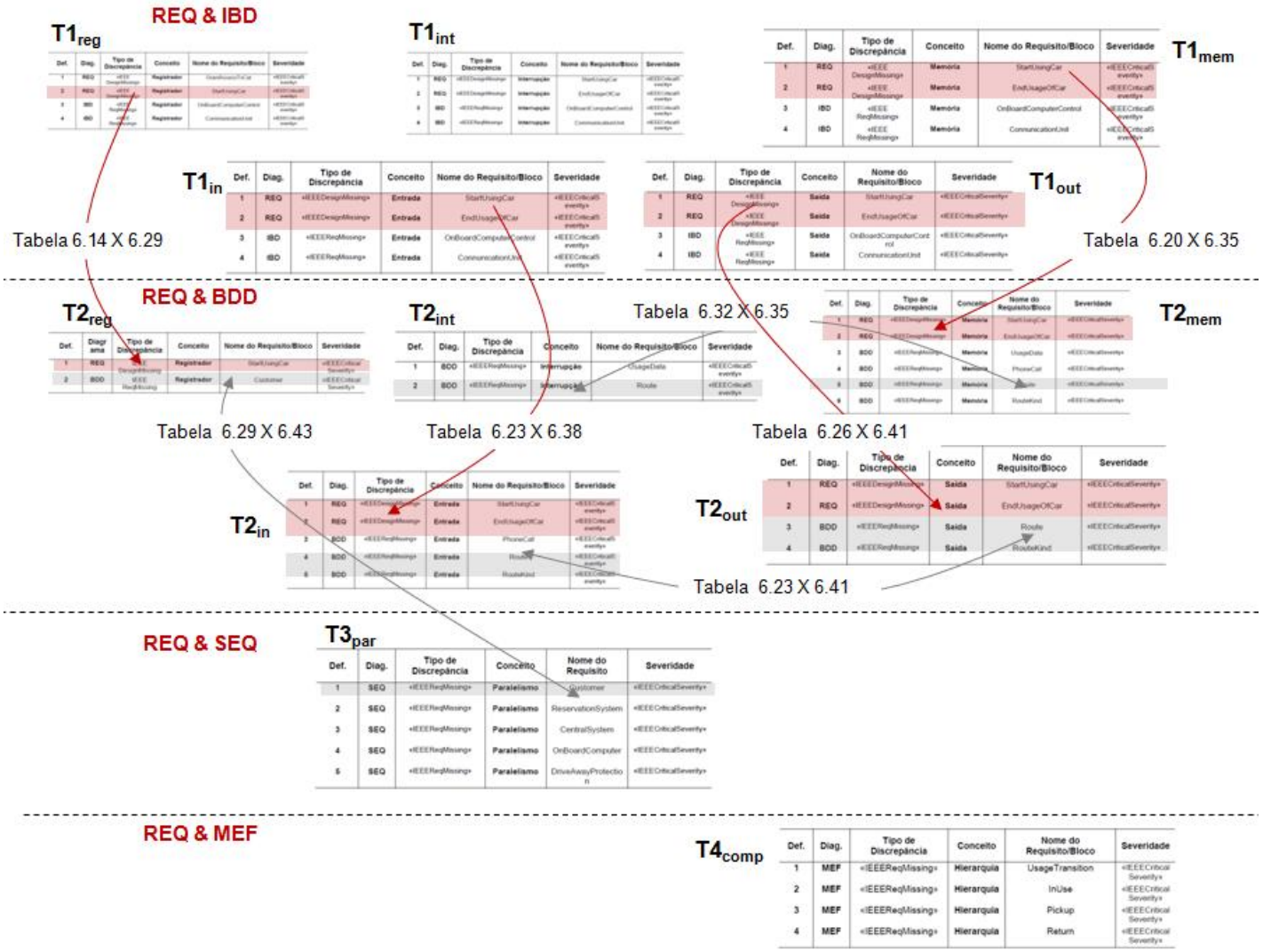


Figura 6.26 - Propagação de defeitos entre as fases do processo SYSMOD.

Na Figura 6.26 é possível observar as propagações de defeitos destacadas com setas/rótulos das respectivas tabelas apresentadas anteriormente nesta seção. Por exemplo, a propagação de defeitos do diagrama IBD para o diagrama BDD (vide transição Tabela 6.14 X 6.29, Técnica $T1_{reg}$ X $T2_{reg}$) ilustra que um mesmo defeito foi propagado de uma fase de desenvolvimento do processo SYSMOD para outro. Em particular, essa propagação ocorreu entre as fases de especificação de requisitos e contexto de sistema do processo SYSMOD (vide Figura 3.10, Capítulo 3).

Em geral, após a aplicação da técnica T1 foi possível constatar defeitos que levam os diagramas REQ e IBD a se tornarem inconsistentes. Por exemplo, a Técnica $T1_{int}$ e $T1_{reg}$ detectaram defeitos pertinentes à garantia de acesso ao sistema bem como de inicialização e uso do veículo da agência de aluguel. Esses defeitos se não forem corrigidos, podem levar a um mau funcionamento do sistema de computador de bordo, em particular permitindo acesso e uso de veículos por pessoas não autorizadas. Além disso, um defeito comum observado em todas as técnica diz respeito ao bloco "OnBoardComputerControl", nesse caso sugerindo uma carência de especificação no diagrama REQ sobretudo com respeito a operação do computador de bordo.

6.4 VANT Tiriba

Esse exemplo explora a aplicação da Técnica de Leitura $T5_{cond}$na especificação de um Veículo Aéreo Não Tripulado utilizado em um trabalho de mestrado orientado por uma pesquisadora que participa do projeto INCT-SEC. Tal especificação trata de parte da especificação arquitetural da VANT Tiriba (SILVA, 2012). Essa especificação completa pode ser encontrada nos trabalhos do Grupo de Engenharia de Software da Universidade Estadual de Maringá (UEM). Essa especificação inclui a descrição em termos de dois diagramas – Diagrama Interno de Blocos e Modelo Simulink. Dessa forma, a técnica $T5_{cond}$ aqui descrita utiliza os artefatos ilustrados na Figura 6.26 e Figura 6.27 respectivamente.

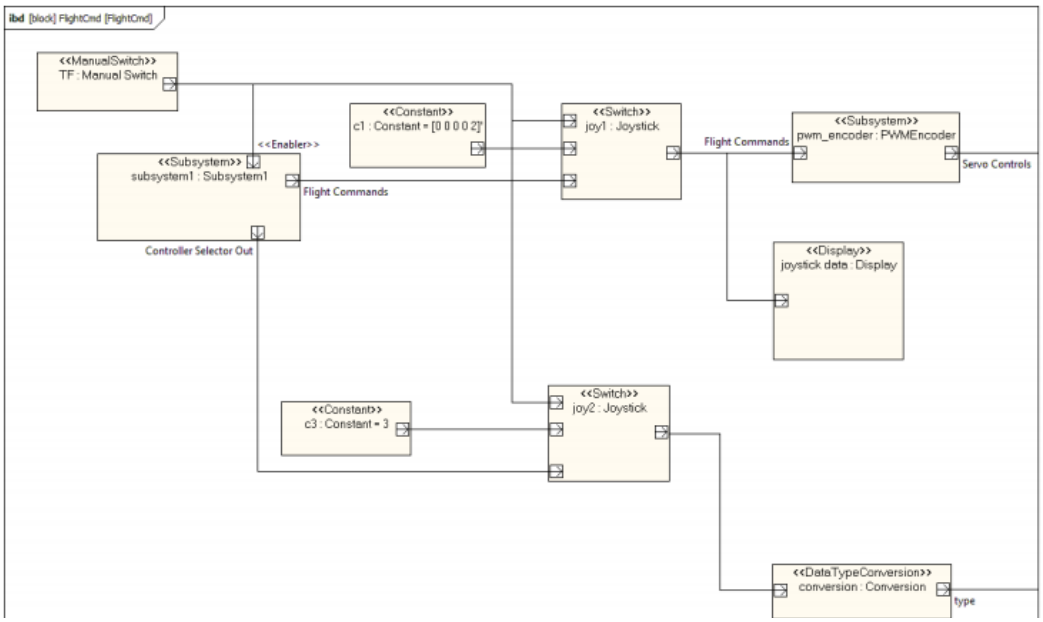


Figura 6.27 - Trecho do subsistema da VANT Tiriba em IBD/SysML (SILVA, 2012).

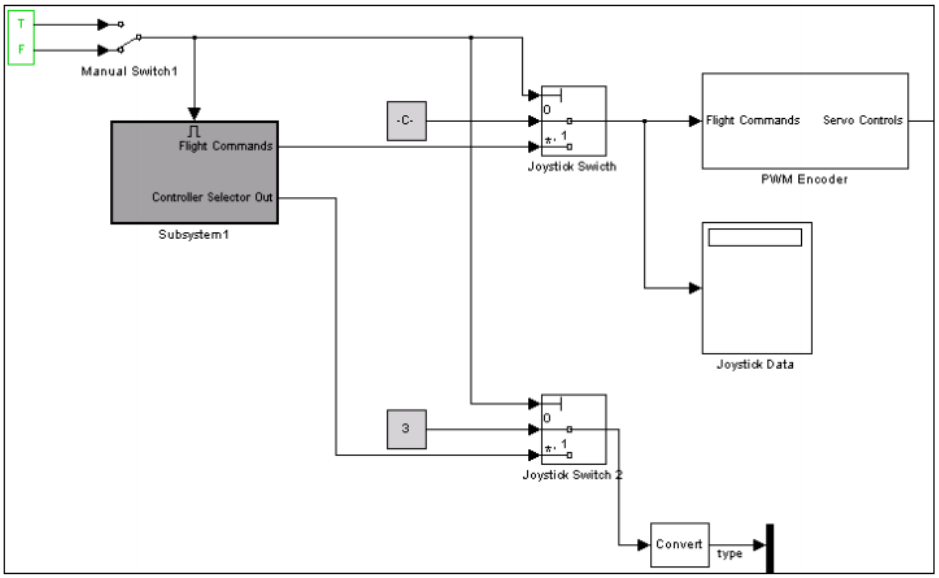


Figura 6.28 - Trecho do subsistema da VANT Tiriba em Simulink (SILVA, 2012).

a) Técnica de Leitura $T5_{cond}$(IBD, SML)

Como exposto no Capítulo 3, a técnica $T5_{cond}$ explora a identificação de defeitos relativos a estruturas condicionais conforme a cobertura requerida pela DO-178C. Os artefatos de entrada para essa técnica são os diagramas: IBD e o Simulink (SML).

Etapa I – Preparação do IBD

Na primeira etapa da técnica, são realizadas as devidas marcações com auxílio dos estereótipos «DO178Input» e «DO178Gate», como indicado nas Diretrizes da Etapa 1 da Técnica $T5_{cond}$. Observando o diagrama IBD (Figura 6.21) foi possível realizar as marcações (✓) mostradas na Tabela 6.47:

Tabela 6.47 - Marcações realizadas no diagrama IBD após aplicação da Etapa I da Técnica $T5_{cond}$.

Blocos	«DO178Input»	«DO178Gate»
ManualSwitch	✓	
Joystick1		✓
Joystick2		✓

Etapa II – Preparação do SML

Análogo à Etapa I, foram feitas as marcações com os estereótipos «DO178Input» e «DO178Gate». Observando o diagrama SML (Figura 4.16) foi possível realizar as marcações (✓) mostradas na Tabela 6.48.

Tabela 6.48 - Marcações realizadas no diagrama SML após aplicação da Etapa II da Técnica $T5_{cond}$.

Blocos	«DO178Input»	«DO178Gate»
ManualSwitch	✓	
Joystick1		✓
Joystick2		✓

Etapa III – Comparação entre o diagrama IBD e SML

Na terceira etapa, são realizadas as verificações comparando os Diagramas IBD e SML simultaneamente com respeito aos estereótipos «DO178Input» e «DO178Gate» previamente anotados, como indicado nas Diretrizes da Etapa 3 da técnica $T5_{cond}$. Dessa forma, a Tabela 6.49, retrata os requisitos e/ou blocos com detectados com defeitos.

Tabela 6.49 - Parte do Formulário de Discrepância após aplicação da Técnica $T5_{cond}$.

Def.	Diag.	Tipo de Discrepância	Conceito	Nome do Requisito/Bloco	Severidade
1	SML	«IEEEDesignMissing»	Condição	Joystick1	«IEEEMajorSeverity»
2	SML	«IEEEDesignMissing»	Condição	Joystick2	«IEEEMajorSeverity»

Em suma, após a aplicação da Técnica $T5_{cond}$ foram observados dois defeitos no diagrama IBD conforme apresentados na Tabela 6.49. Tais defeitos são relativos ao conceito de condição, uma vez que eles estão associados com a cobertura da norma internacional DO178C. Além disso, tais defeitos foram detectados na comparação do Simulink com o IBD (Etapa 3), que procura avaliar se elementos de estruturas condicionais do Simulink estão devidamente representadas no IBD. Assim, tais defeitos foram classificados como sendo «IEEEMajorSeverity», uma vez os blocos com defeitos não descrevem apropriadamente a operação de "joystick" da VANT Tiriba, como por exemplo, o comportamento "default" da porta lógica "switch" existente no Simulink não foi encontrado, de maneira consistente, no diagrama IBD.

6.5 Considerações Finais

Neste capítulo foram apresentados três exemplos para avaliação das técnicas RTSS no que diz respeito à capacidade de detectar defeitos em diagramas SysML e diagramas Simulink. Tais estudos foram relatados passo a passo, explicando como cada uma das técnicas de leitura deve ser aplicada nos pares de artefatos que ela recebe como entrada.

Como mencionado em outros capítulos, pelo fato de considerar-se importante a adoção de um processo para o desenvolvimento de software em geral, inclusive para SEs, as técnicas dão suporte à inspeção de artefatos gerados ao longo do processo SYSMOD. No entanto, elas também podem ser aplicadas mesmo que um processo não tenha sido adotado, bastando que haja os artefatos de entrada da técnica.

Assim, embora os exemplos aos quais se teve acesso não possuíssem todos os artefatos, mesmo que o processo SYSMOD estivesse sendo utilizado, as técnicas foram aplicadas sempre que pertinente. Tais Exemplos de aplicação foram: (i) "Computador de Bordo" que utilizou o exemplo do livro (WEILKIENS, 2008), que adotou o SYSMOD; (ii) "APE", indicado pelo autor Tim Weilkiens, que embora tivesse adotado o SYSMOD, não possuía todos os artefatos do processo; e "VANT-Tiriba", que não adotou o processo SYSMOD.

Os principais comentários a serem destacados desses estudos são os seguintes:

- As técnicas foram capazes de detectar defeitos nos artefatos abordados por elas, quer sejam defeitos próprios de uma fase de desenvolvimento, quer sejam defeitos vindos de fases anteriores – isso pode mostrar a utilidade das técnicas;
- Defeitos identificados por uma determinada técnica foram também identificados por técnicas que devem ser aplicadas em momentos subsequentes, evidenciando que, casos elas tivessem sido aplicadas à medida que os artefatos são construídos, provavelmente esses defeitos não teriam se propagado para as outras fases – isso pode resultar uma diminuição de retrabalho e de custo de correção de defeitos em fases avançadas de desenvolvimento, os quais foram inseridos em fases anteriores;
- Embora todas as técnicas estejam divididas em subpartes relacionadas aos conceitos associados com as normas de certificação ("reg", "int", "mem", "in", "out", "cond") e com os elementos da SysML ("comp", "par"), percebeu-se que os defeitos relacionados a alguns conceitos são mais fáceis de serem identificados em determinadas fases do processo, dado o nível de abstração abordado no artefato de entrada da técnica.

Salienta-se que um risco à validade dos exemplos é o fato deles terem sido conduzidos pelo autor deste trabalho. Esse risco não pode ser contornado, pois se encontrou grande dificuldade em conseguir exemplos para que eles fossem conduzidos.

Capítulo 7
CONCLUSÃO

Neste trabalho foi apresentado o conjunto de técnicas de leitura RTSS (*Reading Technique for SysML and Simulink*) baseadas em Normas de Certificação para Sistemas Embarcados e na estrutura das linguagens SysML e Simulink. Os elementos de certificação considerados nas técnicas foram extraídos de padrões internacionais como DO-178C e UL-98.

Considerando-se a tese inicialmente definida no Capítulo 1 e tendo em vista o exposto anteriormente nos Capítulos 3, 4, 5 e 6, pode-se concluir que, de fato, essas técnicas são capazes de detectar defeitos em diagramas SysML e Simulink, à medida que estes são construídos, evitando a propagação de defeitos de fases iniciais do desenvolvimento para fases posteriores.

Dessa forma, retomando-se o problema apontado no Capítulo 1 de que no contexto de SEs, em geral, os desenvolvedores costumam se a desenvolver a partir de diagramas (modelos) próximos do código, este trabalho contribui com um conjunto de técnicas de leitura para inspeção inseridas ao longo do processo SYSMOD. Dessa forma, o desenvolvedor que desejar adotar um processo pode encontrar neste trabalho a sugestão de uso do processo SYSMOD (já existente na literatura) permeado com atividades de garantia de qualidade de software.

Vale salientar que as técnicas de leitura foram organizadas tomando-se como referência o processo de desenvolvimento SYSMOD, uma vez que esse processo contempla as principais fases de desenvolvimento de software e, de acordo com a literatura, ele tem sido usado no contexto de SEs (GONTE et al., 2005 e SYSMOD, 2013 e WEILKIENS, 2008). No entanto, mesmo que o processo SYSMOD não seja utilizado, as técnicas de leitura aqui definidas podem ser utilizadas para apoiar atividades de garantia de qualidade sempre que os artefatos de entrada das técnicas estiverem disponíveis.

As técnicas RTSS foram definidas considerando-se as seguintes características: (i) normas de certificação, uma vez que quando visto sob o aspecto industrial, os Sistemas Embarcados precisam ser homologados segundo os critérios estabelecidos por essas normas. Assim, teve-se a preocupação de se anteciparem defeitos que são investigados no nível de código para o nível dos diagramas, como por exemplo, defeitos relacionados a interrupções, controladores de CPUs, memórias, entradas/saídas e estruturas de decisão. Isso levou à adoção das normas de certificação internacionais UL-98 (para software embarcado em geral) e DO-178C (para Veículos Aéreos) na definição das técnicas de leitura; (ii) possíveis defeitos de transcrição sintática e semântica entre pares diagramas da linguagem SysML. Nesse caso, as técnicas verificam a consistência entre os diagramas REQ e IBD, REQ e BDD, SD e REQ, e finalmente REQ e MEF. Também foram explorados defeitos de transcrição de informação entre o diagrama IBD e a linguagem alvo Simulink; (iii) necessidade em estabelecerem-se

marcadores para serem usados nas avaliações individuais dos artefatos de entrada da técnica. Nesse caso, as técnicas RTSS também utilizam um conjunto de estereótipos definidos a partir da taxonomia IEEE STD 1044:2009, e que servem para anotar (marcar) os artefatos inspecionados bem como classificar os defeitos encontrados.

Finalmente, em decorrência da necessidade de sistematização das técnicas, para que todas elas fossem definidas de forma padronizada e sistemática, estabeleceu-se um processo que foi extraído da experiência em elaborar a primeira técnica do conjunto RTSS. Esse processo promoveu a criação padronizada e organizada das demais técnicas e pode servir como inspiração para a definição de outros processos que visem outros tipos de técnicas de leitura.

Para avaliação das técnicas RTSS, foram realizados dois experimentos controlados e três estudos à medida que as técnicas foram elaboradas. Assim, a partir dos resultados obtidos foi possível constatar os seguintes pontos:

- A primeira técnica definida, e que serviu de referência para a definição das outras técnicas da família RTSS, estava consistentemente formulada. Ela estava redigida de forma clara e objetiva, os estereótipos usados eram de fato úteis para se fazer as anotações (marcações) nos artefatos inspecionados e eles caracterizavam corretamente os defeitos encontrados nos artefatos. Além disso, pode-se perceber que ambos os formatos de escrita das técnicas RTSS – formatos texto e de fluxograma– são igualmente adequados na detecção de defeitos, o que sugere que qualquer um dos dois formatos pode ser utilizado sem que ocorra interferência na efetividade de aplicação da técnica;
- Na prática as técnicas com diretrizes condicionais menores eram manipuladas ao longo da atividade de inspeção de forma mais apropriada. Com base nessa constatação, a definição das técnicas de leitura da família RTSS foi definida em subpartes;
- Em se tratando da eficiência, a técnica $T4_{comp}$avaliada apresentou tempo razoável de aplicação, uma vez que tal tempo foi inferior ao encontrado em relatos tradicionais da literatura;
- Em geral, as técnicas RTSS, quando utilizadas ao longo de um processo de desenvolvimento, como é o caso do SYSMOD, propiciam o suporte a atividades de verificação e validação, de tal forma que defeitos não sejam propagados para etapas de desenvolvimento futuras (i.e., mais próximas do código fonte);
- Todos os defeitos relativos a cada um dos conceitos (registradores, interrupção, memória, entrada/saída e de elementos condicionais) das normas UL-98 e DO178C, puderam ser observados nos exemplos de aplicação mostrando, portanto, a viabilidade de uso de normas de certificação para SEs;
- As técnicas também promovem a identificação de defeitos, em particular associados com os elementos sintáticos e semânticos da linguagem SysML, como concorrência e hierarquia;
- Nos exemplos conduzidos – VLT/APE, Computador de Bordo e VANT-Tiriba – caso as técnicas de leitura RTSS fossem aplicadas à medida que os diagramas eram concluídos, os defeitos detectados poderiam ser corrigidos tão logo quanto possível, evitando-se a eventual propagação de defeitos e, consequentemente, re-trabalho futuro e possível aumento no custo de desenvolvimento.

7.1 Contribuições e Limitações

Considerando-se as lacunas de pesquisa existentes na área de SEs — apontadas ao longo do Capítulo 2 — notadamente para o uso de Atividades de Verificação & Validação (V&V) ao longo de um processo de desenvolvimento as principais contribuições são:

- Um conjunto de técnicas de leitura para diagramas SysML e Simulink, as quais contemplam atributos de certificação de SEs segundo a UL-98 e a DO-178C; atributos dos diagramas SysML como concorrência e hierarquia; e atributos da taxonomia IEEE STD 1044:2009;
- A sistematização do processo realizado para a elaboração das técnicas;
- Diretrizes de como aplicar as técnicas de leitura para que elas possam compor uma abordagem de garantia de qualidade de software para o processo SYSMOD; e
- Estereótipos como forma para anotar (marcar) os artefatos que são utilizados nas etapas de preparação e comparação pertinentes as técnicas de leitura.

Como limitações deste trabalho vale considerar os seguintes pontos:

- As normas de certificação consideradas neste trabalho estão restritas ao contexto de SEs. Eventualmente, normas de certificação para outros domínios poderiam levar à identificação de outros tipos de defeitos nos diagramas aqui considerados; e
- Durante a avaliação das técnicas de leitura, constatou-se uma dificuldade em se localizar e ter acesso a diagramas de domínio público produzidos em SysML e Simulink. Além disso, houve dificuldade ao acesso a diagramas SysML gerados a partir do processo de desenvolvimento SYSMOD. Tais dificuldades levaram a restrições operacionais, sobretudo na definição dos exemplos e avaliação experimental decorrente do uso de tais diagramas. Na tentativa de minimizar esse problema, o autor do processo SYSMOD foi contatado e, gentilmente, liberou os diagramas do exemplo APE, esclarecendo que, apesar de ter vários outros sistemas modelados com base nesse processo, os documentos eram sigilosos.

7.2 Lições aprendidas

Com base no aprendizado adquirido durante este trabalho, algumas lições podem ser destacadas:

- Para a produção de técnicas de leitura, especialmente para o contexto de SEs, é importante delimitar o escopo de aplicação das técnicas. Assim, é importante a definição clara das metas a serem alcançadas com destaque para os tipos de defeitos que se desejam investigar.
- A elaboração de técnicas de leitura para SEs é uma atividade multidisciplinar, pois envolve as áreas de Engenharia de Software e Sistemas Embarcados, uma vez que na primeira encontram-se várias iniciativas para outros domínios de aplicação e, na segunda, identificam-se os problemas pertinentes à área de SEs;

- A organização das técnicas de leitura em três partes distintas (Etapa I, Etapa II e Etapa III), assim como foi feito em outros trabalhos de engenharia de software, permite que o inspetor siga a técnica mais fielmente. Dessa forma, os artefatos de entrada são inspecionados isoladamente, para que depois eles possam ser comparados com o objetivo de identificar defeitos que possam ter ocorrido pela evolução do processo de desenvolvimento.
- Técnicas de leitura devem ser escritas de forma a contemplar os seguintes aspectos: (a) simplicidade de leitura para execução da atividade de inspeção; (b) concisão no uso de termos para evitar ambiguidade e subjetividade na inspeção; e (c) tamanho da técnica para a manipulação da mesma durante a inspeção.
- A facilidade em conduzir atividades de inspeção no contexto de SEs (assim como em outras áreas) envolve um mínimo de conhecimento tanto em Engenharia de Software como na área específica.

7.3 Publicações Decorrentes do Trabalho

Em decorrência deste trabalho foram publicados artigosà medida que os resultados eram alcançados. A seguir, segue a listagem com as contribuições decorrentes deste trabalho em ordem cronológica de publicação:

- FREIRE, G.; **ANTONIO, E. A.;** PEDRO, L.; NEPOMUCENO, J.; CAURIN, G.; FABBRI, S. *Applying Reengineering on Simulink Model Assisted by UML statechart*. In: Proceedings of I Conference on Critical Embedded Systems (CBSEC), 2011, pp. 1-6.

- **ANTONIO, E**. A; FREIRE, G.; NEPOMUCENO, J.; PEDRO, L.; CASTANHEIRA, B.; MARTINS, R. M.; FERRARI, F. C.; CAURIN, G.; MALDONADO, J. C.; FABBRI, S. *Síntese de algumas iniciativas paraDefinição de um Processo de Desenvolvimento de Software para Sistemas Embarcados Críticos, no âmbito do INCT-SEC*. Internal Workshop (IW), Relatório Técnico, 2011. Disponível em <http://www2.dc.ufscar.br/~lapes/technical_report/EAA_RT_2011.pdf>.

- **ANTONIO, E. A;** FERRARI, F. C.; FABBRI, S. *A Systematic Mapping of Architectures for Embedded Software*. In: Proceedings of II Conference on Critical Embedded Systems (CBSEC), 2012, pp.18-23.

- **ANTONIO, E. A;**ROVINA, R.; FABBRI, S. *Verification and Validation Activities for Embedded Systems A Feasibility Study on a Reading Technique for SysML Models*. In: Proceedings of 16thInternational Conference on Enterprise Information System (ICEIS), 2014, pp. 1-8.

- **ANTONIO, E. A;** FERRARI, F. C.; CAURIN, G.; FABBRI,.*A Set of Metrics for Characterizing Simulink Model Comprehension*. Journal of Computer Science & Technology, JCS&T, vol 14, no. 2, Oct. 2014.

7.4 Trabalhos Futuros

Como trabalhos futuros destacam-se os seguintes pontos:

- Aprimoramento das técnicas de leitura, para incorporar defeitos relacionados a outras normas de certificação, deixando as técnicas apropriadas para outros domínios;
- Adequação a outros tipos de diagramas, como por exemplo, o diagrama paramétrico da SysML;
- Aprimoramento dos tipos de defeitos tratados pelas técnicas de leitura, uma vez que nem todos os aspectos das normas aqui utilizadas foram contemplados;
- Aprimoramento das palavras-chave (termos) utilizado pelas taxonomias visando a melhora da cobertura dos defeitos identificados pelas técnicas de leitura;
- Condução de estudos experimentais para avaliação das técnicas RTSS por indivíduos de diferentes perfis;
- Continuidade à investigação das métricas definidas para diagramas Simulink, com o objetivo de poder usá-las para avaliar a qualidade de tais diagramas isoladamente; e
- Levantamento e produção, junto à comunidade de SEs, de um repositório de domínio público dos diagramas SysML e Simulink. Para fins de experimentação.

REFERÊNCIAS

ANTONIO, E. A. **Desenvolvimento de um WebLab Para Estudo e Caracterização de Sistemas WDM**. São Paulo, SP: Universidade Presbiteriana Mackenzie, 2008. Mestrado.

ANTONIO, E. A.; FERRARI, F.; FABBRI, S. A Systematic Mapping of Architectures for Embedded Software. II Conference on Critical Embedded Systems (CBSEC). **Anais**... Campinas, Brazil: IEEE, 2012. p. 1–6.

ANTONIO, E. A.; FREIRE, G.; PEDRO, L.; et al. **Síntese de algumas iniciativas para Definição de um Processo de Desenvolvimento de Software para Sistemas Embarcados Críticos, no âmbito do INCT-SEC**. São Carlos, Brazil: [s.n.], 2011.

ANTONIO, E. A.; ROVINA, R.; FABBRI, S. C. P. F. (EDS.). Verification and Validation Activities for Embedded Systems: A Feasibility Study on a Reading Technique for SysML Models. ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems. **Anais**... Lisbon, Portugal: SciTePress, 2014. v. 1, p. 8.

ARCURI, A.; IQBAL, M. Z.; BRIAND, L. Black-box system testing of real-time embedded systems using random and search-based testing. Proceedings of the 22nd IFIP WG 6.1 international conference on Testing software and systems. **Anais**... ICTSS'10. Berlin, Heidelberg: Springer-Verlag, 2010. p. 95–110. ISBN 3-642-16572-9, 978-3-642-16572-6.

AURUM, A.; PETERSSON, H.; WOHLIN, C. State-of-the-Art: Software Inspections after 25 years. **Software Testing Verification and Reliability**, [S.l.], 2002. v. 12, n. 3, p. 133–154.

MBSE. Model-based System Engineering Project. **APE: Active Phase Experiment**, 2011.Disponível em <http://mbse.gfse.de/>. Acesso em 31 de jun. 2014.

BASILI, V.; CALDIERA, G.; LANUBILE, F.; SHULL, F. **Studies on Reading Techniques**. [S.l: s.n.], 1996.

BASILI, V.; CALDIERA, G.; ROMBACH, D. The goal question metric approach. **Encyclopedia of Software Engineering**. [S.l.]: Wiley, 1994. . Acesso em: 23 mar. 2012.

BASILI, V.; GREEN, S.; LAITENBERGER, O.; SHULL, F.; ZELKOWITZ, M. V. **The Empirical Investigation of Perspective-Based Reading**. [S.l: s.n.], 1996.

BELGAMO, A. **GUCCRA: Técnicas de Leitura para construção de Modelos de Casos de Uso e Análise de Documento de Requisitos**. São Carlos, Brazil: [s.n.], 2005.

BELGAMO, A.; FABBRI, S.; MALDONADO, J. C. TUCCA: improving the effectiveness of use case construction and requirement analysis. Empirical Software Engineering, 2005. 2005 International Symposium on. **Anais**... [S.l: s.n.], 2005. p. 10 pp.

BUESCHER, T. W.; WILKINSON, R. T. Requirements modeling for real-time software development. Aerospace and Electronics Conference, 1990. NAECON 1990., Proceedings of the IEEE 1990 National. **Anais**... [S.l.]: IEEE, 1990. p. 613–617 vol.2.

CU, C.; JEPPU, Y.; HARIRAM, S.; MURTHY, N. N.; APTE, P. R. A new input-output based model coverage paradigm for control blocks. 2011 IEEE Aerospace Conference. **Anais**... [S.l.]: IEEE, 2011. p. 1–12. ISBN 978-1-4244-7350-2.

DANIELS, D. Thoughts from the DO-178C committee. System Safety, 2011 6th IET International Conference on. **Anais**... [S.l: s.n.], 2011. p. 1–7.

DENGER, C.; CIOLKOWSKI, M. High Quality Statecharts through Tailored, Perspective-Based Inspections. EUROMICRO Conference. **Anais**... Los Alamitos, CA, USA: IEEE Computer Society, 2003. v. 0, p. 316.

DESAI, M. **UL 1998 - Software in Programmable Components**. North Carolina Laura Elan: Underwriters Laboratories Inc., Research Triangle Park, 1998. Disponível em: <http://www.ul.com/global/documents/offerings/industries/hightech/software/UL_software1998.pdf>. Acesso em jun. 2014.

DUBOIS, H.; PERALDI-FRATI, M.; LAKHAL, F. A Model for Requirements Traceability in a Heterogeneous Model-Based Design Process: Application to Automotive Embedded Systems. 2010 15th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS). **Anais**... [S.l.]: IEEE, 2010. p. 233–242. ISBN 978-1-4244-6638-2.

EBERT, C.; JONES, C. Embedded Software: Facts, Figures, and Future. **Computer**, [S.l.], Abr 2009. v. 42, n. 4, p. 42–52. ISSN 0018-9162.

ELAMKULAM, J.; GLAZBERG, Z.; RABINOVITZ, I.; et al. Detecting design flaws in UML state charts for embedded software. Proceedings of the 2nd international Haifa verification conference on Hardware and software, verification and testing. **Anais**... HVC'06. Berlin, Heidelberg: Springer-Verlag, 2007. p. 109–121. ISBN 978-3-540-70888-9.

FERNANDEZ-LOPEZ, M.; GOMEZ-PEREZ, A.; JURISTO, N. METHONTOLOGY: from Ontological Art towards Ontological Engineering. Proceedings of the AAAI97 Spring Symposium. **Anais**... Stanford, USA: [s.n.], 1997. p. 33–40.

GONTE, F.; YAITSKOVA, N.; DERIE, F.; et al. APE: the Active Phasing Experiment to test new control system and phasing technology for a European Extremely Large Optical Telescope. In: GRUNEISEN, M. T.; GONGLEWSKI, J. D.; GILES, M. K. (Eds.). [S.l: s.n.], 2005. p. 58940Z–58940Z–12.

GRAAF, B.; LORMANS, M.; TOETENEL, H. Embedded software engineering: the state of the practice. **IEEE Software**, [S.l.], Dez 2003. v. 20, n. 6, p. 61– 69. ISSN 0740-7459.

GUILHERME FREIRE; LEONARDO PEDRO; ERIK ANTONIO; et al. Applying Reengineering on Simulink Model Assisted by UML Statechart. Conference on Critical Embedded Systems (CBSEC). **Anais**... São Carlos, Brazil: IEEE, 2011. p. 1–6.

HATEBUR, D. **A Pattern- and Component-Based Process for Embedded Systems Development**. Faculty of Engineering Department of Computer Science Software Engineering, 2006.

HAYHURST, K.; VEERHUSEN, D. S.; CHILENSKI, J. J.; RIERSON, L. K. **A Practical Tutorial on Modified Condition/Decision Coverage**. [S.l: s.n.], 2001.

HEISEL, M.; HATEBUR, D. A model-based development process for embedded systems. Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme. **Anais**... [S.l: s.n.], 2005. p. 191.

HESSEL, A.; LARSEN, K. G.; MIKUCIONIS, M.; et al. Testing Real-Time Systems Using UPPAAL. [S.l: s.n.], 2008. v. 4949/2008, p. 77–117.

IEEE. IEEE Standard Classification for Software Anomalies. **IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)**, [S.l.], 2010. p. 1–23.

IEEE Standard Glossary of Software Engineering Terminology. **IEEE Std 610.12-1990**, [S.l.], 1990. p. 1.

JACKSON, M. **Problem frames: analyzing and structuring software development problems**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. ISBN 0-201-59627-X.

KANDT, R. K. Experiences in Improving Flight Software Development Processes. **IEEE Software**, [S.l.], Jun 2009. v. 26, n. 3, p. 58–64. ISSN 0740-7459.

KARSAI, G.; NARAYANAN, A. Composition of Embedded Systems. Scientific and Industrial Issues. Lecture Notes in Computer Science. [S.l.]: Springer Berlin / Heidelberg, 2007. v. 4888, p. 1–18. ISBN 978-3-540-77418-1.

KEHTARNAVAZ, N.; GOPE, C. DSP System Design Using Labview and Simulink: A Comparative Evaluation. 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. **Anais**... [S.l.]: IEEE, 2006. v. 2, p. II–II. ISBN 1-4244-0469-X.

KO, K.; KANG, K. C. **ASADAL/PROVER: A Toolset for Verifying Temporal Properties of Real-Time System Specifications in Statechart**. [S.l: s.n.], 1999.

LEE, E. A. **Actor-Oriented Design: A focus on domain-specific languages for embedded systems**. ,22 Jun 2004. [S.l: s.n.]. Disponível em: <http://chess.eecs.berkeley.edu/pubs/362.html>. Acesso em 31 de jun. 2014.

LIGGESMEYER, P.; TRAPP, M. Trends in Embedded Software Engineering. **IEEE Softw.**, [S.l.], Mai 2009. v. 26, n. 3, p. 19–25. ISSN 0740-7459. . Acesso em: 23 mar. 2012.

MAGICDRAW. **MAGICDRAW: Architecture Made Simple**. ,7 Abr 2013. [S.l: s.n.]. Disponível em: <http://www.nomagic.com/>. Acesso em 31 de jun. 2014.

MARUCCI, R. A.; MALDONADO, J. C.; TRAVASSOS, G. H.; FABBRI, S. C. P. F. OORTs/ProDeS: Definição de Técnicas de Leitura para um Processo de Software Orientado a Objetos. I Simpósio Brasileiro de Qualidade de Software. **Anais**... Gramado: [s.n.], 2002. Disponível em: <http://www.lbd.dcc.ufmg.br:8080/colecoes/sbqs/2002/008.pdf>.

MARWEDEL, P. **Embedded System Design:Embedded Systems Foundations of Cyber-Physical Systems**. 2.ª ed. [S.l: s.n.], 2011. ISBN 978-94-007-0256-1. . Acesso em: 23 mar. 2012.

MATHWORKS. **Simulink - Simulation and Model-Based Design (2012a)**. Disponível em: <http://www.mathworks.com/products/simulink/>. Acesso em 31 de jun. 2014.

MATHWORKS. **Aerospace and Defense**. ,4 Abr 2013. [S.l: s.n.]. Disponível em: <http://www.mathworks.com/aerospace-defense/index.html>. Acesso em 31 de jun. 2014.

MELLEGAARD, N.; STARON, M. Characterizing model usage in embedded software engineering: a case study. Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. **Anais**... ECSA '10. New York, NY, USA: ACM, 2010. p. 245–252. ISBN 978-1-4503-0179-4. Disponível em: <http://doi.acm.org/10.1145/1842752.1842800>. Acesso em: 23 mar. 2014.

MENKHAUS, G.; ANDRICH, B. Metric Suite for Directing the Failure Mode Analysis of Embedded Software Systems. In Proc. of ICEIS. **Anais**... [S.l: s.n.], 2005.

MISHRA, P. Processor validation: a top-down approach. **IEEE Potentials**, [S.l.], Mar 2005. v. 24, n. 1, p. 29– 33. ISSN 0278-6648.

MYERS, G. J.; SANDLER, C. **The Art of Software Testing**. [S.l.]: John Wiley & Sons, 2004. ISBN 0471469122.

NAVIDI, W. **Statistics for Engineers and Scientists**. 3.ª ed. New York: McGraw-Hill, 2010. ISBN 978-0073376332.

NOERGAARD, T. **Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers**. [S.l.]: Newnes, 2005. ISBN 0750677929.

OBERMAISSER, R.; KOPETZ, H. (EDS.). **GENESYS: An ARTEMIS Cross-Domain Reference Architecture for Embedded Systems**. [S.l.]: Suedwestdeutscher Verlag fuer Hochschulschriften, 2009. ISBN 3838110404.

OMG. **OMG Systems Modeling Language : The Official OMG SysML site (2010)**. Disponível em: <http://www.omgsysml.org/>. Acesso em: 23 mar. 2012.

OMG. **UML Profile for MARTE: Modelling and Analysis of Real-Time Embedded Systems v 1.1 (2011a)**. Disponível em: <http://www.omg.org/spec/MARTE/1.1>. Acesso em: 23 mar. 2012a.

OMG. **UML: Unified Modeling Language, Infrastructure - v2.4.1**. Disponível em: <http://www.omg.org/spec/UML/2.4.1/>. Acesso em: 23 mar. 2012b.

OMG. **UML: Unified Modeling Language, Superstructure - v2.4.1**. Disponível em: <http://www.omg.org/spec/UML/2.4.1/>. Acesso em: 23 mar. 2012c.

PASTOR, O.; MOLINA, J. C. **Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling**. [S.l.]: Springer, 2007. ISBN ISBN 978-3-540-71867-3. Disponível em: <http://www.springer.com/computer/swe/book/978-3-540-71867-3>. Acesso em: 25 mar. 2012.

PENTTI, H.; ATTE, H. Failure Mode and Effects Analysis of software-based automation systems. VTT Industrial Systems, STUK-YTO-TR 190. **Anais**... [S.l: s.n.], 2002. p. 190.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic Mapping Studies in Software engineering. Bari, Italy: Blekinge Institute of Technology, 2008. p. 71–80.

POST, H.; SINZ, C.; KAISER, A.; GORGES, T. Reducing False Positives by Combining Abstract Interpretation and Bounded Model Checking. 23rd IEEE/ACM International Conference on Automated Software Engineering, 2008. ASE 2008. **Anais**... [S.l.]: IEEE, 2008. p. 188–197. ISBN 978-1-4244-2187-9.

PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**. 7.ª ed. [S.l.]: McGraw-Hill Higher Education, 2009. ISBN 0073375977.

RAUSCH, A.; BARTELT, C.; TERNITÉ, T.; KUHRMANN, M. The V-Modell XT Applied - Model-Driven and Document-Centric Development. Proceedings 3rd World Congress for Software Quality. **Anais**... [S.l: s.n.], 2005. ISBN 3-9809145-3-4.

SABHARWAL, S.; SINGH, S. K.; SABHARWAL, D.; GABRANI, A. An event-based approach to generate test scenarios. 2010 International Conference on Computer and Communication Technology (ICCCT). **Anais**... [S.l.]: IEEE, 2010. p. 551–556. ISBN 978-1-4244-9033-2.

SAMPATH, P.; RAJEEV, A. C.; RAMESH, S.; SHASHIDHAR, K. C. Testing Model-Processing Tools for Embedded Systems. 13th IEEE Real Time and Embedded Technology and Applications Symposium, 2007. RTAS '07. **Anais**... [S.l.]: IEEE, 2007. p. 203–214. ISBN 0-7695-2800-7.

SHULL, F. J. **Developing Techniques for Using Software Documents: A Series of Empirical Studies**. Maryland: University of Maryland, 1998.

SILVA, R. F. DA. **SyMPLES: Uma Abordagem de Desenvolvimento de Linha de Produto para Sistemas Embarcados baseada em SysML**. [S.l.]: Universidade Estadual de Maringá (UEM), 2012.

SIMON, D. E. **An Embedded Software Primer**. 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 020161569X.

SINHA, A.; SMIDTS, C. HOTTest: A model-based test design technique for enhanced testing of domain-specific applications. **ACM Trans. Softw. Eng. Methodol.**, [S.l.], Jul 2006. v. 15, n. 3, p. 242–278. ISSN 1049-331X. . Acesso em: 23 mar. 2012.

SPARX. **Enterprise Architect 10 Full Lifecycle UML Modeling Software**. ,1 Abr 2013. [S.l: s.n.]. Disponível em: <http://www.sparxsystems.eu/>. Acesso em: 1 abr. 2013.

SYSMOD. **System Modelling Process**. Disponível em: <http://www.sysmod.de/>. Acesso em: 1 abr. 2013.

TEKINERDOGAN, B.; SOZER, H.; AKSIT, M. Software architecture reliability analysis using failure scenarios. **J. Syst. Softw.**, [S.l.], Abr 2008. v. 81, n. 4, p. 558–575. ISSN 0164-1212. . Acesso em: 23 mar. 2012.

TEWARI, A. **Modern Control Design: With MATLAB and SIMULINK**. [S.l.]: John Wiley & Sons, 2002. ISBN 0471496790.

TRAVASSOS, G. H.; SHULL, F.; CARVER, J. **A Family of Reading Techniques for OO Design Inspections**. [S.l: s.n.], 2000.

TRAVASSOS, G. H.; SHULL, F.; CARVER, J.; BASILI, V. R. **Reading Techniques for OO Design Inspections**. [S.l: s.n.], 1999.

TRAVASSOS, G. H.; SHULL, F.; FREDERICKS, M.; BASILI, V. R. Detecting defects in object oriented designs: Using reading techniques to increase software quality. In Conference on Object-oriented Programming Systems, Languages & Applications (OOPSLA. **Anais**... [S.l: s.n.], 1999. p. 47–56.

UNDERWRITERS LABORATORIES. **UL-98 Standards**. Disponível em: <http://www.ul.com/>. Acesso em: 30 set. 2013.

WEILKIENS, T. **Systems Engineering with SysML/UML: Modeling, Analysis, Design**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008. ISBN 0123742749, 9780123742742.

ZAMBONI, A. B.; THOMMAZO, A.; HERNANDES, E.; FABBRI, S. C. P. F. StArt Uma Ferramenta Computacional de Apoio à Revisão Sistemática. Salvador BA: [s.n.], 2010. p. 1–12.

Apêndice A

PLANEJAMENTO DOS MAPEAMENTOS SISTEMÁTICOS

Neste Apêndice é apresentado o modelo de GQM (*Goal, Question, Metrics*) utilizado para nortear o desenvolvimento deste trabalho bem como a condução dos Mapeamentos Sistemáticos realizados.

A Figura 7.1 ilustra o GQM planejado para a condução dos Mapeamentos Realizados e apresentados no Capítulo 2 deste trabalho. O modelo do GQM foi estruturado a partir da definição inicial da tese, seguindo pelos objetivos a serem alcançados bem como das principais questões de pesquisa a serem investigadas. Vale destacar que algumas métricas para avaliação dos dados coletados (a partir dos Mapeamentos Sistemas) foram propostas e aparecem no final do modelo GQM.

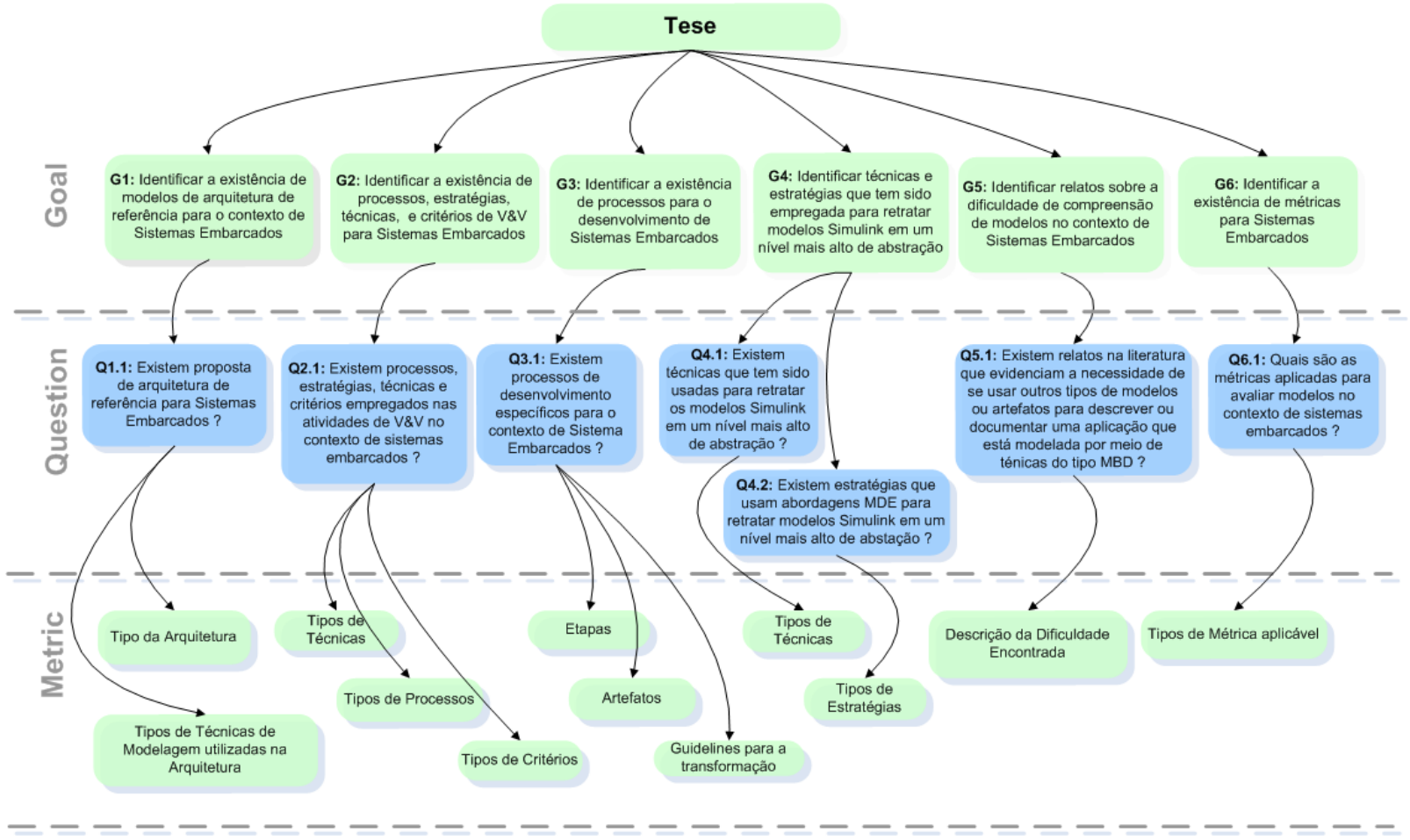


Figura 7.1 - Modelo GQM elaborado para planejar os mapeamentos sistemáticos.