Tratamento de Requisitos Não Funcionais em Sistemas Ciberfísicos com a Utilização de Orientação a Aspectos

Rafael Antônio Vitalli[1], Sidnei Renato Silveira[1], Edison Pignaton de Freitas[2], Cristiano Bertolini[1]

[1]Departamento de Tecnologia da Informação
Universidade Federal de Santa Maria (UFSM) - Campus Frederico Westphalen – RS

[2]Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS) – Porto Alegre – RS

{rafavitalli,sidneirenato.silveira,edisonpf,cbertolini}@gmail.com

Resumo. *Sistemas ciberfísicos possuem condições de comportamento e restrições definidos como requisitos não funcionais. Esses requisitos se caracterizam por afetar vários componentes de maneira irregular e por se distribuírem pelo sistema. Como uma solução a esse problema surgiu o paradigma de orientação a aspectos. Neste contexto, este estudo propõe a realização de um estudo de caso, a partir da aplicação do paradigma de orientação a aspectos, por meio da metodologia RT-FRIDA (Real-Time - From RequIrements to Design using Aspects) adaptada para a especificação de sistemas ciberfísicos, com o objetivo de apresentar os benefícios da utilização do paradigma de orientação a aspectos no tratamento de requisitos não-funcionais. A avaliação da utilização da metodologia é apresentada aplicando métricas de separação de conceitos, acoplamento, coesão e tamanho para aferir a qualidade do projeto.*

Palavras-chave: *Engenharia de software, sistemas ciberfísicos, requisitos não funcionais, paradigma de orientação a aspectos.*

Abstract. *Cyber-physical systems have behavioral conditions and restrictions defined as non-functional requirements. These requirements are characterized by affecting multiple components in an irregular manner and distribute themselves through the system. As a solution to this problem arose the policy paradigm aspects. In this context, this study suggests carrying out a case study with the application of aspect-oriented paradigm, by RT-FRIDA methodology (Real-Time - From requirements to design using Aspects) adapted to the specification of cyber-physical systems, to present the benefits of using the aspect orientation paradigm in the treatment of non-functional The evaluation of the use of the methodology is presented applying metrics of separation of concerns, coupling, cohesion and size to assess the quality of the project.*

Keywords: *Software engineering, cyber-physical systems, non-functional requirements, aspect-oriented paradigm.*

1. Introdução

Os avanços tecnológicos das últimas décadas têm levado a uma nova geração de sistemas, os sistemas ciberfísicos (CPS – *Cyber-Physical Systems*). Os sistemas ciberfísicos fazem parte de uma área emergente e que trata das próximas gerações de serviços inteligentes que estarão no mercado em diversas áreas, como saúde, transporte, automação industrial, construção e espaços inteligentes (ZHANG, 2012). Os CPS são aplicados nas mais diversas áreas, como em veículos, por meio de sistemas automotivos avançados, como o de anticolisão, na indústria, na medicina, com dispositivos médicos altamente confiáveis, sistemas militares, controle e segurança de tráfego, geração e distribuição de energia, entre outros (LEE, 2015). A relação entre elementos computacionais e físicos tem avançado, aumentando consideravelmente a capacidade de adaptação, autonomia, eficiência, funcionalidade, confiabilidade, segurança e facilidade no uso de sistemas ciberfísicos.

Sistemas ciberfísicos, como outros sistemas, possuem condições de comportamento e restrições nas funções oferecidas referentes ao tempo, processo de desenvolvimento, padrões e qualidades globais, definidos como requisitos não (SOMMERVILLE, 2007).

Introduzir requisitos não funcionais em software é uma tarefa complexa, geralmente por se apresentarem muito vagos, repercutirem em todo o sistema e pela dificuldade de serem expressos em elementos unitários (tais como classes, seus atributos e métodos) oferecidos pelo paradigma de orientação a objetos. Tais requisitos não apresentam uma localidade onde é possível isolá-los (FREITAS, 2007).

O processo de desenvolvimento de sistemas ciberfísicos envolve diversas atividades importantes do processo de software, tais como a elicitação, análise e especificação de requisitos, além de resolução de conflitos e validação, denominada Engenharia de Requisitos (ER). A ER tem como objetivo principal especificar claramente os requisitos dos *stakeholders*, fazendo com que os Engenheiros de Software adquiram um melhor entendimento sobre as funcionalidades, restrições e propriedades do sistema a ser desenvolvido, assim como do ambiente em que este estará inserido (SOMMERVILLE, 2007).

O desenvolvimento de software orientado a aspectos (AOSD – *Aspect-Oriented Software Development*) tem a intenção de facilitar a manutenção e o reuso de sistemas, abordando a técnica de separação de interesses (SOMMERVILLE, 2007).

Vantagens como manutenção, reuso e adaptabilidade motivam a pesquisa sobre desenvolvimento de software orientado a aspectos em sistemas ciberfísicos, buscando de alguma forma contribuir neste amplo espaço de estudo.

Neste sentido, o principal objetivo deste trabalho é o de apresentar um estudo que evidencie os benefícios na utilização da Orientação a Aspectos (OA) no tratamento de requisitos não funcionais em sistemas ciberfísicos. Para isso, realizou-se a aplicação de uma metodologia de desenvolvimento orientada a aspectos em um estudo de caso, bem como sua avaliação.

Neste contexto, este trabalho está organizado da seguinte forma: a Seção 2 apresenta os principais conceitos das áreas que envolvem este estudo, destacando-se

sistemas ciberfísicos, requisitos não funcionais, orientação a aspectos e métricas de avaliação; a Seção 3 apresenta trabalhos correlacionados, abordando a análise de requisitos na orientação a aspectos. A Seção 4 apresenta a metodologia utilizada no desenvolvimento deste trabalho. A Seção 5 apresenta o estudo de caso realizado. Na Seção 6 é apresentada a avaliação do uso da metodologia com a utilização de métricas. Finalizando o artigo, são apresentadas as considerações finais e as referências empregadas.

2. Referencial Teórico

Esta seção apresenta um referencial teórico sobre as principais áreas que envolvem este estudo, destacando-se sistemas ciberfísicos, requisitos não funcionais, orientação a aspectos e métricas de avaliação.

2.1. Sistemas Ciberfísicos (CPS – *Cyber-Physical Systems*)

Um sistema ciberfísico é a interação da computação com o mundo físico, por meio de sensores e atuadores, cujas operações são monitoradas, coordenadas, controladas e integradas por um dispositivo computacional (LEE; SESHIA, 2014).

Os CPS também podem ser definidos como sistemas físicos, biológicos e de engenharia cujas operações são integradas, monitoradas e controladas por um núcleo computacional. Os componentes são ligados em rede em todas as escalas. O núcleo computacional é um sistema embarcado, geralmente exige resposta em tempo real, e é na maioria das vezes distribuído (GILL, 2011).

A geração anterior à dos sistemas ciberfísicos é geralmente conhecida como sistemas embarcados, e encontraram aplicações em áreas diversas, no entanto, tendem a focar mais nos elementos computacionais, enquanto que sistemas ciberfísicos enfatizam o papel das ligações entre os elementos computacionais e elementos físicos (KHAITAN; McCALLEY, 2014).

Com a integração entre sensores, redes e componentes de computação com o controle da física, acredita-se que os CPS transformarão a maneira de interagir e manipular o mundo físico (LEE; SESHIA, 2014).

Alguns exemplos que indicam essa necessidade são: a qualidade, segurança e eficiência em infraestruturas críticas, como a geração, transporte e distribuição de energia, água e gás; a integração e otimização do controle de tráfego; cuidados de saúde realizados, cada vez mais, por meio de automatização avançada, integrando dispositivos inteligentes, além de fornecer acesso seguro a registros médicos eletrônicos (SANISLAV; MICLEA, 2012).

CPS são diferentes de sistemas tradicionais embarcados/tempo real, das atuais redes de sensores sem fio ou de sistemas *desktop*, já que possuem certas características que os diferenciam: 1) estreita integração entre computação e processos físicos; 2) o software está incorporado em todo o sistema embarcado ou componente físico e os recursos são geralmente limitados; 3) a rede, podendo ser com/sem fio, WLAN (*Wireless Local Area Network*), *Bluetooth*, GSM (*Global System for Mobile Communications*), entre outras, forma sistemas distribuídos com dispositivos variados; 4) componentes complexos em espaço e tempo; 5) capacidade de adaptação e

reorganização/reconfiguração; 6) alto grau de automação, com uso de avançadas tecnologias de controle de *feedback*; 7) confiabilidade nas operações (SHI *et. al*, 2011).

2.2. Requisitos Não Funcionais

Os requisitos de um sistema, segundo Sommerville (2007), podem ser divididos em requisitos funcionais, não funcionais e de domínio. Requisitos funcionais definem as funcionalidades que o sistema deve fornecer, ou seja, os serviços, as reações e o seu comportamento. Os requisitos não funcionais (RNF) se referem às restrições que afetam os serviços e funções do sistema, tais como restrições de tempo, utilização de recursos ou padrões de desenvolvimento. Estão relacionados também às propriedades do sistema como confiabilidade, desempenho e disponibilidade e não diretamente às funcionalidades específicas oferecidas pelo sistema. Requisitos de domínio se referem às características e as restrições do domínio da aplicação, como tipos e tamanhos de dados.

Sommerville (2007) classifica os RNF em: 1) requisitos de produto, que especificam o comportamento do produto; 2) requisitos organizacionais, ligados às políticas e procedimentos da organização, e 3) requisitos externos, que abrangem requisitos originados de fatores externos e a interação com outros sistemas.

Diversas outras classificações podem ser encontradas na literatura, como a proposta por Bertagnolli (2004), utilizada na metodologia FRIDA (*From RequIrements to Design using Aspects*) para gerenciar os requisitos do projeto e auxiliar na identificação e tratamento de requisitos não funcionais aplicando o paradigma orientado a aspectos, como mostra a Figura 1.

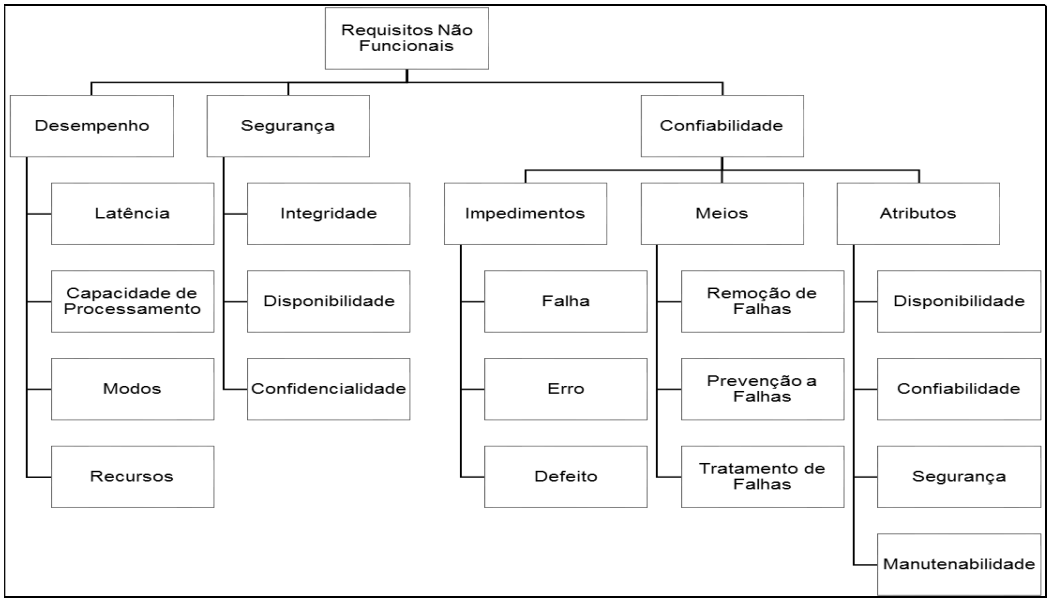

Figura 1. Classificação de RNFs proposta em Bertagnolli (2004)

Freitas (2007), baseando-se na metodologia FRIDA, propôs uma classificação para Sistemas de Tempo-real Embarcados e Distribuídos (STrED), apresentada na Figura 2.

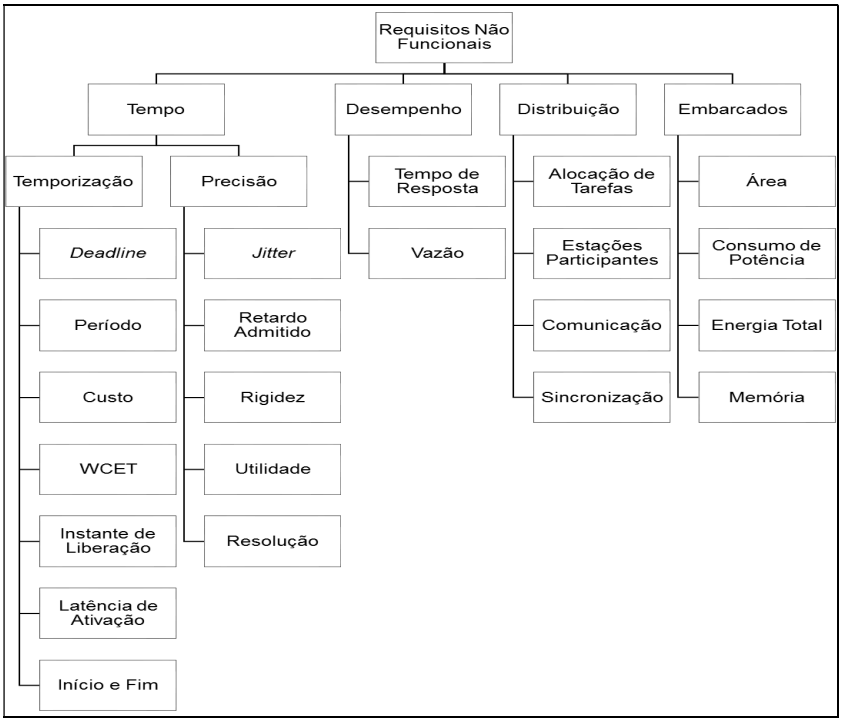

Figura 2. Classificação de RNFs proposta em Freitas (2007)

O desenvolvimento de software requer técnicas que tratam mais do que aspectos relacionados à sua funcionalidade e a descrição de atividades e entidades, mas que consideram aspectos relativos à qualidade, confiabilidade, desempenho, segurança, usabilidade, entre outros, classificados esses como RNFs (BERTAGNOLLI, 2004). Os RNFs, ao contrário dos requisitos funcionais, não expressam nenhuma função a ser realizada pelo sistema, mas restrições sobre os serviços ou as funções oferecidas pelo sistema.

Na maioria das vezes, RNFs são mais importantes que requisitos funcionais individuais, pois se um requisito funcional apresentar uma falha pode comprometer parte do sistema, enquanto que uma falha em cumprir um requisito não funcional pode tornar todo o sistema inútil (SOMMERVILLE, 2007).

2.3. Orientação a Aspectos

A Programação Orientada a Aspectos (AOP – *Aspect-Oriented Programming*) foi proposta por Kiczales *et. al* (1997), como uma solução a alguns problemas da Programação Orientada a Objetos (OOP – *Object-Oriented Programming*).

Para Kiczales *et. al* (1997), as técnicas de OOP não são suficientes para capturar, claramente, todas as decisões importantes na implementação de um software. Neste sentido, a AOP se apresenta como uma técnica de desenvolvimento de software baseada na separação de interesses (*concerns*). Assim, sistemas poderiam ter diferentes interesses transversais, projetados de forma independente, utilizando técnicas de AOP. Cada interesse transversal é chamado de *aspecto*. A orientação a aspectos pode diminuir a complexidade dos modelos, separando seus diferentes interesses. Estes aspectos podem incluir pontualidade, estabilidade, tolerância a falhas e segurança. Técnicas de

AOP podem aumentar a compreensão, adaptabilidade e a capacidade de reutilização do sistema.

Segundo Bertagnolli (2004), Kiczales observa que os atuais métodos e notações concentram-se em encontrar e compor unidades funcionais, as quais são representadas por procedimentos, funções, módulos, objetos, etc. Porém, dessa forma, aspectos não funcionais, que são importantes para preservar a funcionalidade da aplicação, não são tratados. Esse fato é decorrente, principalmente, de algumas limitações das linguagens de programação. O Desenvolvimento de Software Orientado a Aspectos (AOSD – *Aspect-Oriented Software Development*) teve seu início a partir dessa ideia.

Segundo Sommerville (2007), o AOSD é baseado em uma abstração chamada *aspecto*, com a sua introdução na orientação a objetos, representando as *preocupações transversais* do projeto, sendo usado com outras abstrações como objetos e métodos que fazem parte de um sistema.

O AOSD tem, como objetivo principal, a separação de assuntos na estruturação do projeto e do código, com a ideia de separar as *preocupações transversais* da parte funcional do sistema, ou seja, tratar os RNFs separadamente (SOMMERVILLE, 2007).

Alguns requisitos estão presentes em todo o projeto como, por exemplo, o *log* do sistema. Esses requisitos, que atravessam (do inglês, *crosscutting*) as funcionalidades básicas do sistema, são denominados requisitos transversais (KICZALES *et. al*, 1997).

Segundo Freitas (2007), *aspectos* são entidades que tratam os RNFs e permitem a separação de responsabilidades. Essa preocupação com tais requisitos vem sendo focada nas fases iniciais do ciclo de vida do sistema, podendo auxiliar no estudo e esclarecimento dos RNFs, a fim de evitar futuras mudanças no projeto em fases mais avançadas, o que, geralmente se torna oneroso.

2.4. Métricas de Avaliação

Com o objetivo de medir atributos de qualidade no desenvolvimento de sistemas computacionais, métricas de avaliação são usadas tanto em projetos orientados a objetos quanto a aspectos.

Em sistemas orientados a aspectos, uma relação de métricas e um modelo de qualidade são utilizados para a avaliação (SANT´ANNA *et. ak*, 2003). Quatro conjuntos de métricas são utilizados:

1) Métricas de Separação de Conceitos (Preocupações ou Interesses) (MSC): avalia a capacidade de identificar, encapsular e manipular partes de um sistema que são relevantes a determinado conceito, preocupação ou interesse. Enquadram-se nesse grupo as métricas:

   a) Difusão de Conceitos (preocupações ou interesses) por Componentes (DCC);

   b) Difusão de Conceitos (preocupações ou interesses) por Operações (DCO);

   c) Difusão de Conceitos (preocupações ou interesses) por Linhas de Código (DCLC);

2) Métricas de Acoplamento (MA): indicam a força das interconexões entre componentes no sistema. As seguintes métricas pertencem ao grupo:

   a) Acoplamento entre Componentes (AEC);

b) Profundidade da Árvore de Herança (PAH);

3) Métrica de Coesão (MC): a coesão mede a relação de dependência entre elementos internos de um componente. Apenas uma métrica faz parte deste grupo:

a) Falta de Coesão nas Operações (FCO);

4) Métricas de Tamanho (MT): são aquelas que medem o tamanho do projeto e/ou código do sistema. As métricas componentes deste grupo são as seguintes:

a) Tamanho do Vocabulário (TV);

b) Linhas de Código (LC);

c) Número de Atributos (NA);

d) Peso de Operações por Componente (POC).

A união dos fatores compreensão e flexibilidade são responsáveis pela aferição das qualidades de reuso e manutenibilidade dos sistemas orientados a aspectos. O primeiro fator envolve a facilidade de compreensão de um sistema se comparado a outro, tendo maiores chances de ter seus componentes reusados e maior facilidade de realizar manutenções. O segundo fator diz respeito à capacidade de reuso de um componente específico do sistema em outro projeto, além da facilidade de realizar a manutenção em uma determinada parte do sistema sem maiores implicações para o restante do sistema. O modelo de qualidade é apresentado na Figura 3.

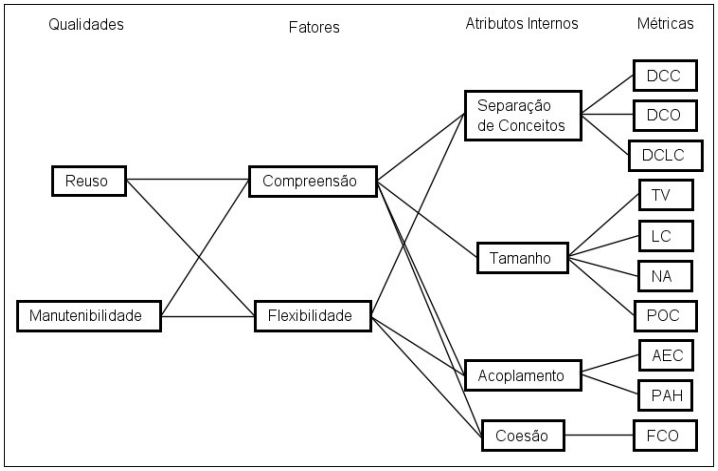

Figura 3. Modelo de Qualidade (SANT´ANNA *et. al*, 2003)

3. Trabalhos Relacionados

Esta seção apresenta alguns trabalhos relacionados ao trabalho aqui apresentado, tendo sido estudados trabalhos envolvendo requisitos não funcionais e o paradigma de orientação a aspectos.

3.1. Orientação a Aspectos Utilizando UML

O modelo de engenharia de requisitos orientado a aspectos foi inicialmente proposto em Rashid (RASHID *et. al*, 2002), com a finalidade de separar interesses transversais na fase de análise de requisitos. Posteriormente este modelo foi adaptado, utilizando o modelo de casos de uso e diagramas em UML (*Unified Modeling Language*),

apresentado em Araújo (ARAÚJO *et. al,* 2002). O modelo de requisitos orientado a aspectos utilizando UML é apresentado na Figura 4.

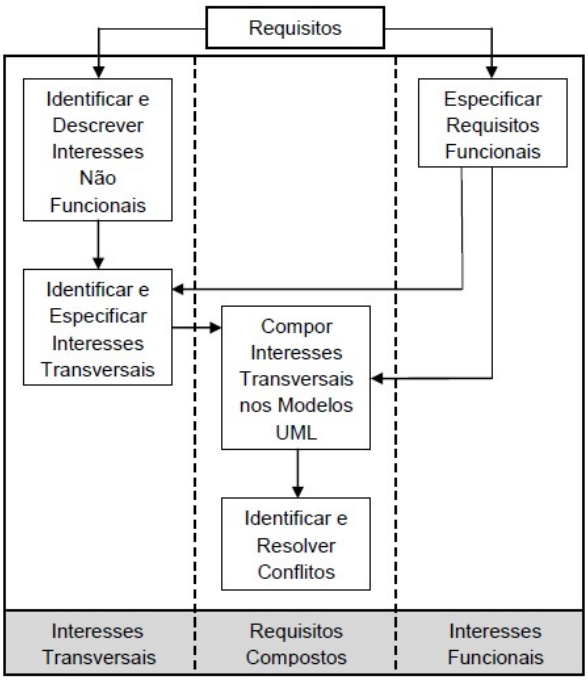

Figura 4. Modelo de requisitos orientado a aspectos de Araújo

(ARAÚJO *et. al,* 2002)

Conforme mostra a Figura 4, o processo é particionado em três etapas. Os requisitos não funcionais são elicitados com a finalidade de identificar quais deles são transversais, considerados como *aspectos*, e que são especificados usando o modelo apresentado no Quadro 1.

Quadro 1. Especificação de interesses transversais (ARAÚJO *et. al*, 2002)

Interesse transversal	<Nome>
Descrição	<Descrição do interesse>
Prioridade	<Prioridade pode ser Máxima, Média e Mínima>
Lista de requisitos	<Requisitos que descrevem o interesse>
Lista de modelos	<Modelos UML influenciados pelo interesse>

Uma análise é realizada, especificando-se os requisitos funcionais, utilizando uma abordagem UML, por meio de um diagrama de casos de uso. Em seguida, um processo denominado análise de requisitos compostos é realizado, compondo requisitos funcionais com os aspectos utilizando modelagem UML, a fim de identificar e resolver possíveis conflitos. Os conceitos de sobreposição, substituição e empacotamento são utilizados nessa etapa.

No relacionamento de *sobreposição* o aspecto modifica o requisito funcional que ele afeta. Neste caso, o aspecto pode ser executado antes ou depois da execução do requisito funcional. Na *substituição*, o requisito de um aspecto substitui o requisito

funcional que ele atravessa. No *encapsulamento*, o requisito de um aspecto encapsula o requisito funcional que ele atravessa. Neste caso, o comportamento descrito pelo requisito funcional é empacotado pelo comportamento descrito pelo interesse transversal.

Um estudo de caso baseado em um sistema de tarifação do tráfego rodoviário é apresentado para evidenciar a aplicação do modelo (ARAÚJO *et. al*, 2002). Nele, motoristas de veículos autorizados são cobrados automaticamente ao passarem por portões de cobrança ao entrar, sair e ao longo da rodovia. Um dispositivo é instalado no veículo do usuário da rodovia, o que permite a sua identificação. No momento que o veículo passar por um dos portões, o sistema deve verificar se está autorizado a trafegar na rodovia e sinalizar com uma luz verde, exibir o valor que será cobrado e realizar a tarifação. No caso do veículo não estar autorizado a transitar na rodovia, ao passar em um dos portões, o sistema deve acender uma luz amarela e fotografar a sua placa.

A descrição do sistema apresentado contém requisitos funcionais e não funcionais. No entanto, a fim de obter uma descrição completa de cada requisito não funcional, é necessário discutir as restrições que o sistema tem de satisfazer. A primeira delas é sobre a ação do sistema no momento em que um veículo passa por um portão de controle. Um diagrama de casos de uso referente ao sistema é apresentado na Figura 5. O interesse transversal não funcional *<<TollGateResponseTime>>* compreende os casos de uso *EnterMotorway*, *ExitMotorway* e *PassSingleToll*, empacotando, assim, o comportamento funcional desses casos de uso.

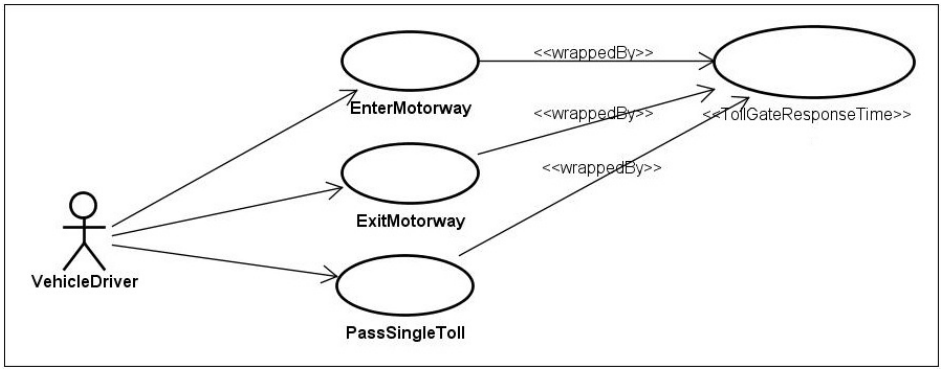

Figura 5. Diagrama de Casos de Uso com interesse transversal (ARAÚJO *et. al*, 2002)

Na composição de interesses transversais podem ocorrer conflitos. Um requisito não funcional é transversal quando afeta mais casos de uso. A partir do modelo apresentado no Quadro 1, o exemplo do requisito não funcional *tempo de resposta* é apresentado no Quadro 2. Nesse exemplo, este requisito não funcional atravessa os casos de uso *PassSingleToll, EnterMotorway* e *ExitMotorway*.

Quadro 2. Exemplo de especificação de um requisito não funcional (ARAÚJO *et. al*, 2002)

Interesse transversal	Tempo de resposta do portão de controle
Descrição	O portão de controle deve reagir antes do motorista deixar a área do portão
Prioridade	Máxima
Lista de requisitos	R1, R2, R3, R4
Lista de modelos	Casos de Uso: Passar Sinalização, Entrar na Rodovia, Sair da Rodovia

Neste exemplo, o conflito entre o interesse transversal de segurança e o de tempo de resposta do portão de controle deve ser analisado, levando-se em conta a influência deles sobre os demais aspectos, a prioridade atribuída inicialmente a cada um e a decisão dos *stakeholders* sobre qual deles priorizar.

3.2. Integração do Modelo NFR *Framework*

Brito e Moreira (2003a) apresentam uma abordagem da engenharia de requisitos com o objetivo de identificar, especificar e integrar preocupações, incluindo interesses transversais. Uma versão estendida desta abordagem foi apresentada em [Brito e Moreira 2003b].

Com a finalidade de ajudar na identificação e especificação de interesses o modelo NFR *Framework* foi integrado ao modelo de engenharia de requisitos [Brito e Moreira 2004]. O modelo genérico é apresentado na Figura 6. O modelo é composto por quatro tarefas principais que são: identificar interesses, especificar interesses, identificar interesses transversais e compor interesses.

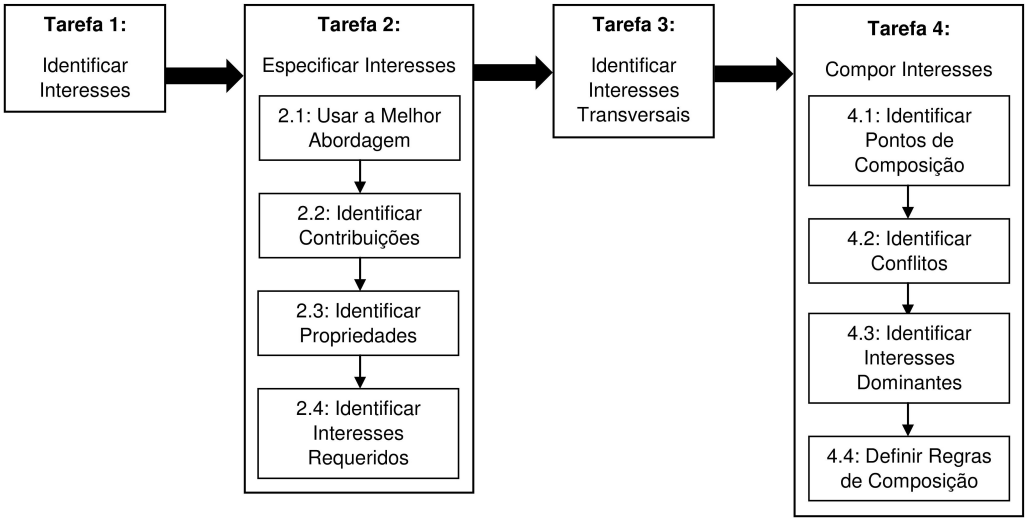

Figura 6. Modelo de separação de interesses adaptado de Brito e Moreira (2004)

O modelo NFR *Framework* é um método de representação e análise de requisitos não funcionais, proposto em (CHUNG *et. al*, 2000).

A tarefa 1, de identificação de interesses, é alcançada por meio da realização de uma completa e exaustiva análise da documentação e quaisquer outras informações fornecida pelos *stakeholders* do sistema. Cada interesse deve ser registrado em um modelo como ilustrado no Quadro 3. Na primeira tarefa são preenchidos os campos *Nome*, *Fonte*, *Stakeholders* e *Descrição*. Os outros campos serão preenchidos durante a tarefa 2.

Quadro 3. Modelo de especificação de interesses (BRITO; MOREIRA, 2004)

Nome	Nome do interesse
Origem	Fonte da informação, por exemplo as partes interessadas, documentos, domínio, catálogos e processos de negócio
Stakeholders	As pessoas que precisam do interesse no sentido de usar o sistema
Descrição	Explica o comportamento desejado do interesse
Classificação	Auxilia a seleção da abordagem mais apropriada para especificar o interesse. Por exemplo: objetivos, funcional, não funcional, *goals*
Contribuição	Representa como um interesse pode ser afetado por outro interesse. Esta contribuição pode ser positiva (+) ou negativa (-)
Prioridade	Manifesta a importância do interesse para as partes interessadas. Pode levar os valores: muito importante, importante, médio, baixa e muito baixa
Interesses Requeridos	Lista dos interesses relacionados ao referido interesse

A tarefa 2, especificar interesses, é dividida em quatro subtarefas:

Tarefa 2.1. Especificar interesses usando a melhor abordagem: pode-se usar qualquer uma das técnicas existentes para especificação de requisitos. Em certos casos, a escolha pode ser feita durante a Tarefa 1, especialmente, se técnicas específicas foram utilizadas no processo de elicitação. Dessa forma, modelos ou outra documentação propostos por essas técnicas devem ser construídos. Esta subtarefa preenche o campo *Classificação* do Quadro 3;

Tarefa 2.2. Identificar contribuições entre interesses: a relação entre dois interesses define a maneira como um interesse afeta o outro. Esta contribuição pode ser colaborativa (positiva +), ou prejudicial (negativa -) entre interesses. Esta subtarefa é adicionada ao campo *Contribuição* do Quadro 3. É com base nesta informação que os conflitos entre os interesses são detectados;

Tarefa 2.3. Identificar prioridades: para cada interesse é necessário identificar o seu grau de importância no sistema. Isso é feito com a ajuda dos *Stakeholders*. Esta informação é adicionada ao campo *Prioridade* do Quadro 3;

Tarefa 2.4. Identificar interesses requeridos: criação de uma lista de interesses que o referido interesse necessita para cumprir o seu papel. Se o interesse não requer outro interesse, usa-se a palavra-chave <nenhum> no campo *Interesses Requeridos* do Quadro 3.

A tarefa 3, identificar interesses transversais, é realizada levando em conta as informações dos campos *Interesses Requeridos* e *Descrição*, do Quadro 3. Um interesse é transversal se for requerido por mais de um interesse.

A tarefa 4, compor interesses, tem o objetivo de compor todas os interesses para que o desenvolvedor possa ter uma ideia de todo o sistema. Quatro subtarefas são utilizadas para orientar a composição:

Tarefa 4.1. Identificar pontos de composição: formado com base nos *Interesses Requeridos* do Quadro 3, identificando os pontos onde a composição é realizada. Isto é melhor ilustrado no Quadro 4, de *Stakeholders versus* Interesses, onde cada célula é preenchida com a lista de interesses transversais (IT) que afetam um determinado interesse e representam um ponto de composição (PC). Uma regra de composição deve ser definida para cada ponto de composição.

Quadro 4. Matriz de identificação de pontos de composição [Brito e Moreira 2004]

	Interesse	...	$Interesse_n$
$Stakeholder_1$	IT_1, IT_2 (IT_A)		IT_2, IT_5 (IT_J)
$Stakeholder_2$	IT_1, IT_3 (IT_B)		
...	...		
$Stakeholder_{nt}$			IT_3, IT_5 (PC_Z)

Tarefa 4.2. Identificar conflitos: esta subtarefa auxilia na identificação de situações de conflito entre os interesses. Para cada ponto de composição é necessário analisar se um dos interesses transversais envolvidos contribui negativamente para qualquer outro, com base no campo *Contribuição* do Quadro 3. Interesses que contribuem positivamente para outros interesses não causam problemas.

Tarefa 4.3. Identificar interesses dominantes: ajuda a resolver os conflitos identificados na subtarefa anterior. Esta subtarefa baseia-se no campo *Prioridade* do Quadro 3. Se a prioridade atribuída a cada interesse for diferente, o problema não é muito difícil de resolver, e o interesse dominante será o de maior prioridade. No entanto, se pelo menos dois interesses transversais possuírem a mesma prioridade, um *trade-off* deve ser realizado entre as partes. A negociação entre as partes se dá propondo a identificação do interesse transversal dominante para um determinado ponto de composição. Se dois ou mais interesses transversais existirem em um determinado ponto de composição, com contribuição negativa e mesma prioridade, os dois primeiros interesses são analisados e identifica-se o dominante. Em seguida, o dominante é analisado com um terceiro interesse e, assim por diante, até que todos os interesses transversais sejam analisados. O resultado é um interesse com a prioridade mais elevada entre todos eles. Em seguida, o segundo interesse transversal dominante é identificado entre os interesses remanescentes e, assim por diante, até que tenhamos uma hierarquia de dependência entre todos os interesses. A identificação do interesse dominante é importante para orientar a regra de composição.

Tarefa 4.4. Definir regras de composição: define a ordem em que os interesses serão aplicados em um ponto de composição, na forma <interesse><operador><interesse>:

- Autorização $I_1 >> I_2$: é uma composição sequencial e significa que o comportamento do interesse 2 começa somente se o comportamento do interesse 1 finalizar com sucesso;

- Desativação $I_1 [> I_2$: é uma composição incapacitante e significa que o comportamento do interesse 2, quando iniciado, interrompe o comportamento do interesse 1;

- Intercalação Pura $I_1 ||| I_2$: é um operador em paralelo e significa que o comportamento do interesse 1 evolui, independentemente, do comportamento do interesse 2;

- Sincronização Pura $I_1 || I_2$: é um outro operador em paralelo e significa que o comportamento do interesse 1 deve ser sincronizado com o comportamento do interesse 2.

3.3. Metodologia FRIDA

A metodologia FRIDA concentra-se na modelagem dos RNFs, na sua análise e decomposição, compreendendo desde a fase de análise até o projeto e início da implementação do sistema. Alguns passos são seguidos pela metodologia para a separação dos RNFs. Cada passo utiliza um ou mais artefatos, denominados unidades de decomposição (BERTAGNOLLI, 2004), como mostra o Quadro 5.

Quadro 5. Unidades de decomposição relevantes em Bertagnolli (2004)

Fase do Modelo	Unidade de Decomposição
Identificação e Modelagem de Requisitos Funcionais	Diagrama de casos de uso e *templates*
Identificação e Modelagem de RNFs	*Checklist*
	Léxico - palavras e expressões
	Matriz de conflitos
Associação dos Requisitos Funcionais com o projeto	Classes e XML
Modelagem Visual de Aspectos	*Template* para aspectos
	Aspectos (estereótipos UML)

Para melhor compreensão, os passos serão descritos a seguir.

Na fase de identificação e modelagem de requisitos funcionais é analisado o problema. Esta fase é responsável por estabelecer os objetivos do projeto, investigar as necessidades do cliente e do domínio da aplicação, além de extrair, analisar, documentar e validar os requisitos. O modelo de casos de uso é adotado para a especificação de requisitos nesta fase. Para documentar cada diagrama de casos de uso é associado a ele

um *template*. Este *template* tem a finalidade de detalhar textualmente cada caso de uso, apresentando informações tais como: identificador, nome, objetivo, autor, atores, prioridade, dentre outros.

Na fase de identificação e modelagem de RNFs foi adotada a utilização de *checklists* como artefato, para realizar a verificação de requisitos, pois se destacam na captura de omissões e oferecer completude para a especificação. Ainda nesta fase, a classificação dos RNFs é apresentada, podendo ser genéricos ou específicos. Os genéricos abrangem os mais abstratos, por exemplo, desempenho, segurança e confiabilidade, enquanto que os específicos resultam da decomposição dos genéricos, formando, assim, uma hierarquia de RNFs.

A fase de identificação e modelagem de RNFs também adotou o uso de artefatos léxicos. Este artefato, denominado processo léxico e descrito em Leite e Oliveira (1995), representa um glossário com palavras-chave relacionadas ao domínio do problema, sendo utilizado para verificar se algum requisito não funcional foi esquecido, mal especificado ou negligenciado pelo desenvolvedor.

A identificação e modelagem de RNFs possui, ainda, a fase de resolução de conflitos. Nesta fase, uma base de conhecimento onde os conflitos, já determinados, entre os RNFs são armazenados, é gerada através de regras que permitem identificar quando dois requisitos encontram-se em conflito. Além desta base, os conflitos são armazenados em uma matriz de conflitos que exibe relacionamentos conflitantes entre RNFs. Após a identificação dos conflitos, a resolução é iniciada baseando-se na estratégia que avalia as prioridades. Em alguns casos, a solução baseada em prioridades não é clara o suficiente para englobar todos os relacionamentos de aspectos de um sistema e a intervenção do engenheiro pode ser necessária para decidir sobre o conflito.

A fase de extração do modelo de análise e projeto consiste em transformar a estruturação dos requisitos em um diagrama de classes. Este modelo compreende um conjunto de conceitos, cujo foco está nos elementos/entidades apropriados ao domínio do problema. Alguns cuidados devem ser tomados nesta fase, pois caso algum RNF tenha sido mapeado no diagrama de classes ele deve ser reavaliado, pois eles devem ser encapsulados em aspectos. Esse mapeamento (RFs para classes e RNFs para aspectos) evita o entrelaçamento e a dispersão de informações por meio de elementos funcionais. Dessa forma, cada classe do diagrama representará somente um requisito funcional.

A fase de associação dos requisitos funcionais com o projeto consiste em empregar o conceito de rastreabilidade, criando um vínculo, em que o(s) diagrama(s) de casos de uso deve(m) estar associado à, pelo menos, uma classe.

A fase da modelagem visual de aspectos consiste na extração de aspectos por meio das informações geradas nas fases anteriores. Cada aspecto é vinculado a um *template* que possui diversas informações, como identificação, aplicabilidade, origem, entre outros. Esta fase é realizada utilizando uma extensão ao diagrama de classes, onde os aspectos são identificados por um rótulo *<<aspect>>* e um nome. Nele, ainda, estereótipos são utilizados definindo seus métodos, *pointcuts*, *advices* e *joinpoints*.

A fase de combinação da visão funcional com a visão não funcional consiste em estabelecer as relações entre as classes e os aspectos, pois esses elementos encontram-se desconectados uns dos outros. Para que essa ligação entre a visão funcional (classes) e a

não funcional (aspectos) seja realizada, deve haver o cruzamento das informações escritas na fase de associação dos requisitos com o projeto, onde a rastreabilidade é aplicada e analisando o *template*, definido na fase de extração do modelo. Dessa maneira, cada classe deve estar associada a um ou vários aspectos, além do que um único aspecto deve existir para cada RNF.

O estereótipo utilizado no relacionamento entre as classes e os aspectos foi o <<*crosscutting*>>, enfatizando que o aspecto é uma propriedade ortogonal à classe que está ligada a ele. Com essa fase conclui-se a modelagem do sistema. E, como última fase, há a geração de código, que consiste na geração da estrutura básica dos elementos do projeto (classes e aspectos).

3.4. Estudo Comparativo

O Quadro 6 apresenta um comparativo entre os trabalhos estudados e a metodologia aplicada no estudo de caso apresentado (FREITAS, 2007).

Quadro 6. Comparativo entre os trabalhos estudados

Item avaliado	Araújo *et. al* (2002)	Brito e Moreira (2004)	Bertagnolli (2004)	Metodologia Utilizada – Freitas (2007)
Fases do ciclo de desenvolvimento do software				
Requisitos		✓	✓	✓
Análise e Projeto	✓	✓	✓	✓
Implementação		✓	✓	✓
Orientação a Objetos	✓		✓	✓
Orientação a Aspectos	✓		✓	✓
Fase de Especificação de Requisitos – Técnicas e Artefatos Adotados				
Casos de uso	✓		✓	✓
Cenários	✓		✓	✓
Diagrama de Sequência do Sistema				
Template para RNFs	✓		✓	✓
Grafo		✓		
Léxico Específico para um Contexto			✓	✓
Base de Conhecimento			✓	✓
Conflitos	✓	✓	✓	✓
Checklist Específico para STrED				✓
Aplicação para Sistemas Ciberfísicos				✓

De acordo com o estudo comparativo realizado, os trabalhos estudados se assemelham em algumas características com a metodologia aplicada neste estudo de caso (FREITAS, 2007), sendo que o ponto principal que a diferencia dos demais é a aplicação de orientação a aspectos para a especificação de sistemas ciberfísicos.

4. Metodologia Utilizada

O desenvolvimento deste trabalho foi realizado com a aplicação da metodologia de desenvolvimento orientada a aspectos RT-FRIDA, adaptada para a especificação de sistemas ciberfísicos a partir de um estudo de caso, focando na separação de requisitos funcionais e não funcionais.

A metodologia RT-FRIDA (*Real-Time* FRIDA), proposta por Freitas (2007), consiste em tratar os requisitos não funcionais de sistemas de tempo-real embarcados distribuídos (STrED), aplicando a orientação a aspectos, sendo adaptada a partir da metodologia FRIDA (*From RequIrements to Design using Aspects*), proposta por Bertagnolli (2004). A escolha da metodologia FRIDA para a adaptação foi pelo fato desta se preocupar com os requisitos não funcionais desde o princípio de vida do sistema, na sua fase de análise, fornecendo ferramentas que facilitam o mapeamento destes requisitos em aspectos, além de apresentar flexibilidade na utilização destas ferramentas.

O objetivo principal da metodologia adaptada RT-FRIDA é a elicitação dos requisitos não funcionais referentes a questões de tempo, distribuição e embarcados, mapeando o seu tratamento em aspectos, por meio da identificação de requisitos com a adaptação de ferramentas da metodologia original e com ideias disponíveis em trabalhos como o de Araújo *et. al* (2002).

Os principais requisitos não funcionais envolvidos em STrED foram levantados e organizados, gerando uma classificação, apresentada no Quadro 7. Em seguida, apresentam-se as três fases principais utilizadas na RT-FRIDA e que correspondem às fases da metodologia original: a) Primeira fase: Identificação e Especificação de Requisitos; b) Segunda fase: Mapeamento de Requisitos em Elementos de Projeto; c) Terceira fase: Projeto do Sistema.

Quadro 7. *Template* para RF de Freitas (2007)

		Item	Descrição
Geral	**Identificação**	**Identificador**	Identificação que permitirá a rastreabilidade do requisito por todo o projeto.
		Nome	Nome do caso de uso associado ao requisito.
		Objetivo	Descrição dos objetivos do caso de uso.
		Autor	Pessoa responsável pela definição.
	Contexto	**Pré-condição**	Estado em que o sistema deve se encontrar antes que este caso de uso possa ser executado.
		Pós-condição	Estado no qual o sistema deve se encontrar após o cenário primário ter sido finalizado.
		Ator primário	Ator considerado a fonte dos eventos que estimulam a execução do cenário primário.
		Ator secundário	Um ator que interage passivamente com o caso de uso, mas não executa nenhuma ação sobre o mesmo.
	Decisão e Evolução	**Prioridade**	Usada para decidir a importância relativa entre os casos de uso. Ela assume um dos seguintes valores: sem prioridade, baixa, média, alta ou crítica.
		Situação	Um requisito pode estar em uma das seguintes situações: 0 - identificado; 1 - analisado; 2 - especificado; 3 - aprovado; 4 -

			cancelado; 5 - finalizado;
Caminhos		**Primário (Normal)**	Descreve o fluxo principal do caso de uso, sem condições de erro, apenas as com resultados positivos.
		Alternativo	Apresenta o fluxo de andamento alternativo para a funcionalidade do caso de uso.
		Excepcional	Apresenta a descrição do fluxo de andamento do caso de uso em uma situação atípica.
Cenários		**Principal**	Descrição dos passos principais que envolvem o cenário onde se insere o caso de uso.
		Variações	Os passos descritos como variações são aqueles que modificam um ou mais passos do cenário principal.

A fase de Identificação e Especificação de Requisitos pode ser subdividida em duas etapas, que são: 1) identificação e especificação de requisitos funcionais: nesta etapa é realizada a identificação das funcionalidades para o sistema e a construção do diagrama de casos de uso, e, ainda, o preenchimento de um *template* detalhando a informação apresentada no diagrama de casos de uso referente aos requisitos funcionais. O Quadro 8 apresenta o referido *template*. A identificação e realização de conflitos é realizada nesta etapa, com o uso de uma matriz onde são relacionados os requisitos funcionais e identificados os requisitos conflitantes; e 2) identificação e especificação de requisitos não funcionais: nesta etapa, uma *checklist* é utilizada para a identificação dos requisitos não funcionais. Nela é elaborado um conjunto de questões especificamente para o domínio de interesse do sistema.

O modelo genérico de uma *checklist* é apresentado no Quadro 8, onde se observa na primeira coluna o espaço reservado às questões de inferência, divididas conforme a classificação apresentada no Quadro 7. O uso deste modelo se dá por meio da resposta aos questionamentos apresentados na primeira coluna e elas definirão a presença ou não do requisito no sistema. Os demais campos sobre a relevância, a prioridade e detalhes sobre condições, restrições ou uma descrição sobre o requisitos são preenchidos.

Quadro 8. Modelo de organização de uma *Checklist* (FREITAS, 2007)

	Relevante	**Prioridade**	**Restrição/Condição/Descrição**
Classificação Genérica			
Classificação Específica			
Pergunta de Inferência			

No estudo foram desenvolvidas quatro *checklists*, uma para cada classe de requisitos não funcionais apresentadas no Quadro 7. O uso deste modelo permite o levantamento de informações como relevância, prioridade, descrição e restrições envolvendo cada requisito.

Após o preenchimento das *checklists*, outro artefato é utilizado para verificar se algum requisito não funcional foi esquecido ou mal especificado. Este artefato, denominado processo léxico é descrito em Leite e Oliveira (1995). O léxico se assemelha a um glossário com palavras-chave relacionadas ao domínio do problema. Para o uso e descrição deste léxico faz-se necessário o uso de uma representação formal.

Neste caso, foi adotada a forma *Backus-Naur Form* (BNF) (BACKUS *et. al*, 1969], que define formalmente uma sintaxe permitindo uma análise estruturada das sentenças do texto descritivo do sistema e dos *templates* de requisitos funcionais, em busca de conceitos não funcionais considerados importantes para o projeto, e que não tenham sido devidamente tratados. Os léxicos criados no estudo de Freitas (2007) encontram-se no Anexo 1.

Obtidas as informações por meio das *checklists* e léxicos, um *template* para cada requisito não funcional é preenchido. O *template* utilizado é apresentado no Quadro 9.

Quadro 9. *Template* para RNF de Freitas (2007)

	Item	Descrição
Identificação	**Identificador**	Identificação que permitirá a rastreabilidade do requisito por todo o projeto.
	Nome	Nome do requisito não-funcional.
	Autor	Pessoa responsável pela identificação e definição.
Especificação	**Classificação**	Classificação a qual pertence o entrelaçamento.
	Descrição	Descrição de como afeta funcionalidades do sistema.
	Casos de Uso Afetados	Lista de casos de uso afetados.
	Contexto	Determina em que momento o entrelaçamento afeta um caso de uso.
	Escopo	(Global/Parcial) O requisito é global se afeta o sistema como um todo, e parcial se afeta apenas uma parte do sistema.
Decisão e Evolução	**Prioridade**	Usada para decidir a importância relativa entre os casos de uso. Ela é assume um dos seguintes valores: sem prioridade, baixa, média, alta ou crítica.
	Situação	Um requisito pode estar em uma das seguintes situações: 0 - identificado; 1 - analisado; 2 - especificado; 3 - provado; 4 - cancelado; 5 – finalizado.

5. Estudo de Caso

Esta seção apresenta um estudo de caso visando demonstrar a aplicabilidade da metodologia RT-FRIDA. O objeto deste estudo de caso é um ventilador pulmonar mecânico para a utilização em anestesias com base em Douglass (1998). O uso do referido estudo de caso se dá com o objetivo de apresentar uma comparação entre a modelagem orientada a objetos apresentada em Douglass (1998) e a modelagem orientada a aspectos realizada neste trabalho.

5.1. Ventilador Pulmonar

De um modo geral, o objetivo dos ventiladores pulmonares é fornecer suporte ventilatório temporário, completo ou parcial a pacientes incapazes de respirar por vias normais devido a fatores como doença, anestesia ou defeitos congênitos (BUTTON, 2002). São usados também para permitir descanso dos músculos respiratórios até que o paciente seja capaz de reassumir a ventilação espontânea.

Os ventiladores pulmonares são classificados de acordo com o ambiente ou as funções primárias para que se destinam, podendo ser Ventiladores de Cuidados Intensivos, Ventiladores Portáteis (ou Ventiladores para *Home Care*) e Ventiladores de

Transporte. Os Ventiladores de Cuidados Intensivos diferenciam-se dos demais, principalmente pela intensidade e evolução da tecnologia, onde integram características complementares com relação aos outros tipos de ventiladores, e que se destinam, em princípio, a melhorar seu desempenho. Fornecem suporte temporário para pacientes críticos, que requerem assistência total ou parcial para manter ventilação adequada, fornecendo trocas gasosas nos pulmões por pressão positiva, abrindo ou mantendo a ventilação dos alvéolos (onde ocorrem as trocas gasosas), e aliviando os músculos ventilatórios até que o paciente estabeleça a ventilação espontânea de forma adequada e segura. Os Ventiladores Portáteis ou Ventiladores para *Home Care* são usados em pacientes que não necessitam de cuidados críticos complexos, mas necessitam de suporte ventilatório por período prolongado. A maioria destes ventiladores é fácil de usar, pois possui controles complexos como os ventiladores de cuidados intensivos. Ventiladores de Transporte devem prover suporte ventilatório no caso de transporte do paciente, sendo projetados para esse fim com características específicas, como dimensões e peso reduzidos, possuir baterias internas e serem usados em intervalos curtos de tempo (FORNAZIER *et. al*, 2011).

Os ventiladores pulmonares evoluíram de simples aparelhos ciclados por pressão, até os sistemas controlados por microprocessadores sofisticados, usados para tratamento de pacientes em estado crítico e com falência respiratória (PEREL; STOCK, 1994).

Na maioria dos procedimentos cirúrgicos de anestesia profunda, o paciente é paralisado, não podendo respirar sem assistência e o ventilador pulmonar é utilizado para esse fim. Durante todo o processo é necessário controlar a concentração de oxigênio e gases anestésicos, denominados gases frescos. Um sistema de fole e campânula é utilizado para controlar a concentração dos gases frescos e empurrar a mistura para o pulmão do paciente. Primeiramente, o fole é preenchido com a mistura de gases que será enviada ao paciente. Com o aumento da pressão sobre o fole, os gases em seu interior são forçados na direção do circuito respiratório pelo ramo inspiratório. A pressão dentro do fole é controlada pelo ventilador e a inspiração é controlada conforme a modalidade de respiração desejada, havendo a separação entre o paciente e o ventilador feita pelo filtro valvular. No momento determinado, o ventilador alivia a pressão no interior da campânula e o fole se distende, diminuindo a pressão no circuito respiratório, ocorrendo a expiração. O fole é preenchido tanto pelos gases frescos quanto por gases expirados pelo paciente, onde existe a presença de CO_2 que não deve ser reenviado ao paciente. Por esse motivo, no próximo ciclo de inspiração o fluxo que sai do fole passa por um filtro de CO_2. Dessa forma, os agentes anestésicos não absorvidos pelo paciente são reaproveitados no próximo ciclo de inspiração (TURRIN, 2011).

A Figura 7 ilustra a entrega de gases frescos ao paciente e a remoção do CO_2.

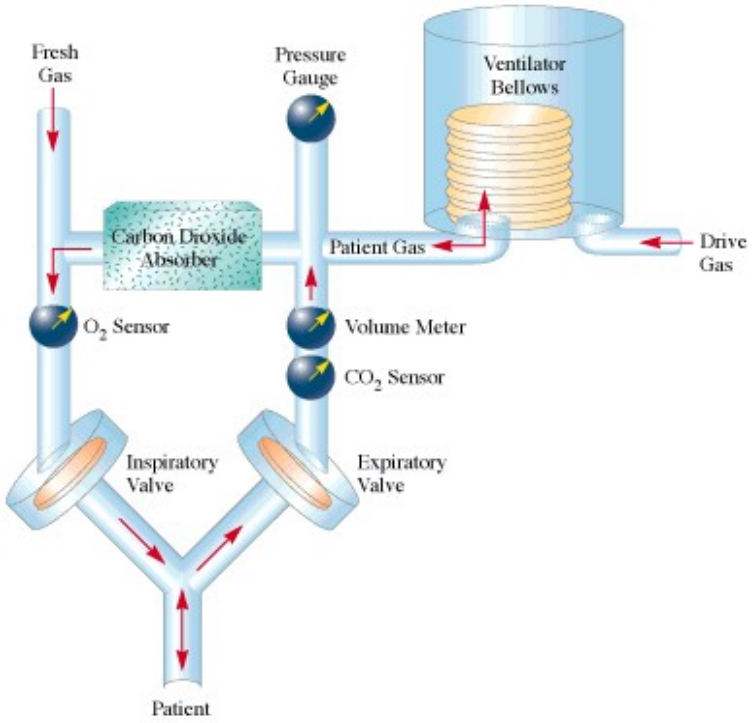

Figura 7. Funcionamento da entrega de gases frescos ao paciente e remoção dos gases residuais em Douglass (1998)

Para a circulação de gases, geralmente os ventiladores executam monitoração para melhorar a segurança do paciente. Os parâmetros mais comuns monitorados são a concentração de O_2 no ramo inspiratório (FiO_2 - Fração Inspirada de Oxigênio), a concentração de CO_2 no ramo expiratório ($etCO_2$ – *End-tidal Carbon Dioxide*), o volume de gás que retorna do paciente com o uso de sensor de fluxo e o sensor de pressão do circuito respiratório, que fornece informações incluindo a relação I:E (relação entre tempo inspiratório e tempo expiratório), a pressão expiratória final e o bloqueio ou obstrução de alguma mangueira que resultará em alta pressão expiratória, e que emitirá um alarme. O ventilador inclui tanto o modo ajustado como o modo por parâmetros medidos (DOUGLASS, 1998).

Os alarmes são classificados em três grupos: informativo (baixa criticidade), cuidado (eventualmente, podem resultar em ferimentos graves ou morte se não forem tomadas medidas corretivas) e crítico (imediata lesão iminente ou morte se não forem tomadas medidas corretivas). Cada um destes alarmes possui um comportamento diferente. Alarmes informativos são exibidos por um breve período de tempo (não mais que dois minutos ou se forem substituídos por alarmes de maior prioridade) e na cor verde. Alarmes de cuidado são exibidos até serem reconhecidos pelo usuário (o usuário pressiona o botão de silêncio do alarme) e na cor amarela. Alarmes críticos são anunciados até o reconhecimento pelo usuário, mas serão reanunciados a cada dois minutos até que a condição do alarme seja resolvida, e são exibidos em cor vermelha (DOUGLASS, 1998)

Os alarmes são exibidos na janela de alarmes, conforme mostra a Figura 8. São exibidos cinco alarmes em ordem de tipo (primeiro os alarmes críticos, seguido por cuidados e depois os informativos). Caso o número de alarmes ativos ultrapasse esse

número, um indicado gráfico é exibido podendo o usuário rolar a janela do alarme para visualizar os demais alarmes.

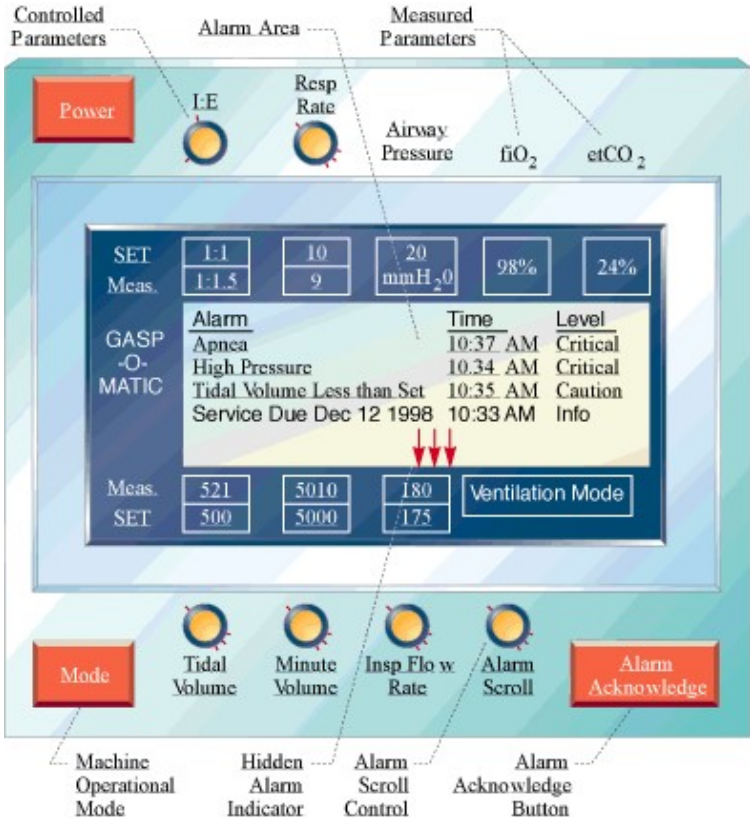

Figura 8. Painel frontal de um ventilador pulmonar apresentado em Douglass (1998)

Conforme mostra a Figura 8, o painel frontal do ventilador fornece todo o controle do usuário. Nele cada parâmetro controlado exibe um valor ajustado pelo usuário e um valor medido pelos sensores no exato momento, com exceção da concentração de oxigênio e dióxido de carbono que são apenas parâmetros medidos.

O ventilador opera em três modos: o modo de ventilação, o modo de configuração e o modo de serviço. O modo de ventilação é o modo de funcionamento normal da máquina, onde ocorre a ventilação do paciente. O modo de configuração permite ao usuário configurar a máquina, para estabelecer os limites dos alarmes e os ajustes dos parâmetros pretendidos. O modo de serviço é usado pelo pessoal de serviço para atualizar o software, calibrar ou substituir os sensores. A seleção do modo operacional é feita com o uso de um botão de três vias no painel frontal.

Não foram considerados, neste estudo de caso, os modos de configuração e de serviço do ventilador pulmonar, mas em um sistema real a sua modelagem exigiria tanto cuidado quanto o modo de ventilação.

A seguir será apresentado o desenvolvimento do estudo de caso de acordo com as fases de identificação e especificação de requisitos, mapeamento de requisitos e projeto do sistema de ventilação do ventilador pulmonar utilizado em anestesias.

5.2. Identificação e Especificação de Requisitos

O primeiro passo na identificação e especificação dos requisitos consiste na extração de suas características funcionais. A partir da breve descrição do sistema, anteriormente apresentada, é possível extrair tais características, ainda que o sistema seja mais complexo que o descrito. Realiza-se o levantamento dos casos de uso do sistema, que são especificados por *templates* de requisitos funcionais (apresentado no Quadro 7).

A segunda etapa consiste na identificação e especificação dos requisitos não funcionais que se encontram na descrição do sistema. Para a identificação, utiliza-se o conjunto de *checklists* apresentado em Freitas (2007). Para identificar requisitos não funcionais, utiliza-se o léxico desenvolvido por Freitas (2007) para STrED na busca de algum requisito que não tenha sido identificado. O léxico encontra-se no Anexo 1.

A próxima etapa consiste na especificação dos requisitos não funcionais. Para esse fim, utiliza-se o *template* de RNFs demonstrado no Quadro 9.

Com a conclusão desta etapa, os requisitos não funcionais (com estereótipo <<*non-functional*>>) são relacionados aos funcionais no diagrama de casos de uso como é possível observar na Figura 9. Para que o diagrama ofereça uma maior clareza, os requisitos não funcionais foram agrupados em pacotes. Para o tratamento destes requisitos foi utilizada uma biblioteca extensível de aspectos de alto nível (DERAF - *Distributed Embedded Real-time Aspects Framework – High Level*) que possui um conjunto de aspectos abstratos e pode ser encontrada em [Freitas et al. 2007].

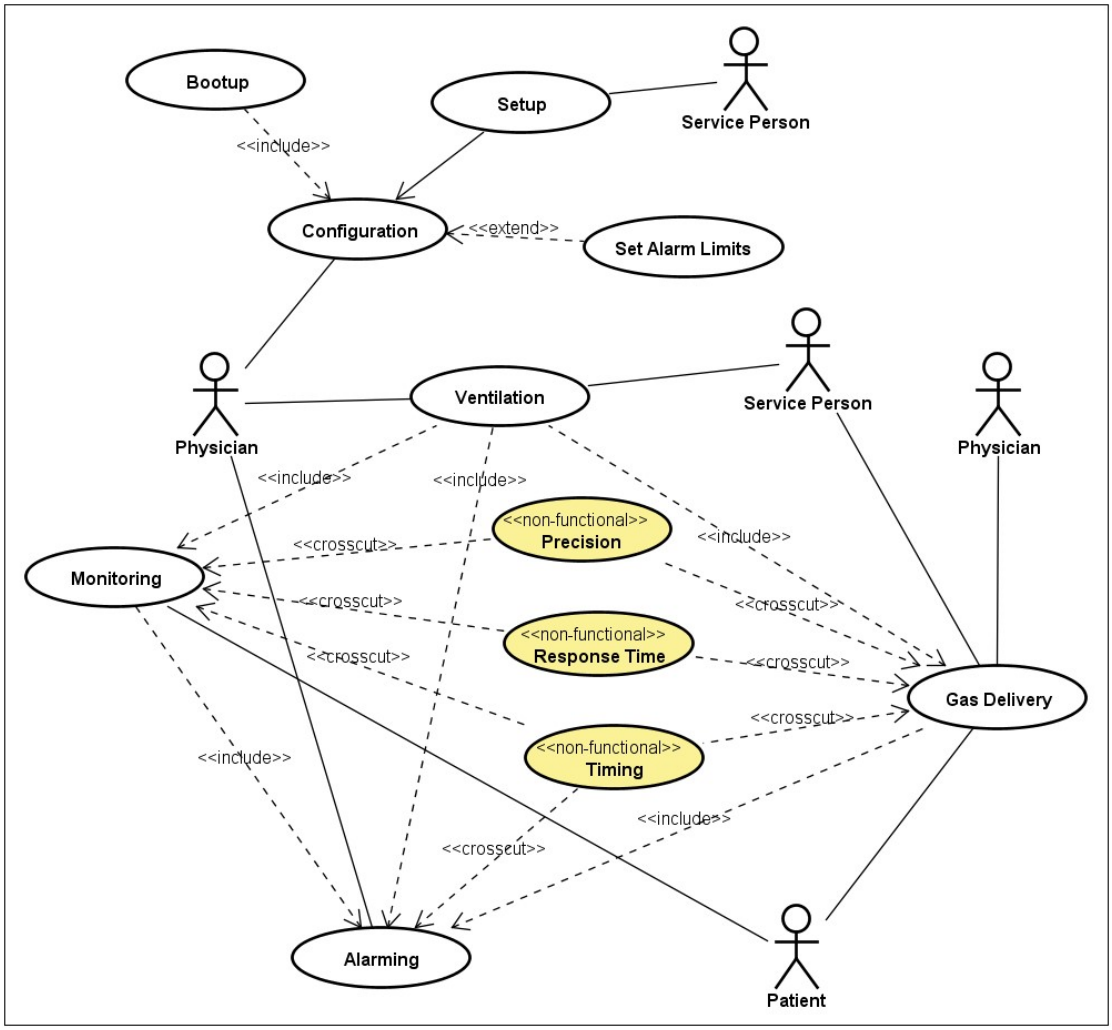

Figura 9. Diagrama de Casos de Uso do ventilador para anestesia adaptado de Douglass (1998) - representação de requisitos funcionais, não funcionais e sua interação

De acordo com a Figura 9, o pacote *Timing* reúne os aspectos que tratam os requisitos de temporização do sistema. O pacote *Precision* reúne os aspectos que tratam os requisitos de precisão relativos ao tempo. O requisito não funcional *Response Time* representa o tempo necessário para que o sistema retorne uma resposta final que dependa da execução de atividades locais ou remotas. Este requisito faz parte da classificação de desempenho e está relacionado tanto ao aspecto temporal quanto de distribuição do sistema. Por esse motivo não está vinculado ao pacote de tempo ou distribuição [Freitas 2007].

5.3. Mapeamento de Requisitos em Elementos de Projeto

O objetivo da fase de mapeamento de requisitos em elementos de projeto é o de realizar o elo entre os requisitos e os elementos de projeto, com a finalidade de evitar o problema de qual elemento de projeto trata determinado requisito, facilitando a manutenção ou evolução do sistema, além do reuso de componentes. Para isso, o mapeamento é realizado, relacionando os requisitos funcionais e as classes que possam atendê-los com os requisitos não funcionais e os aspectos que atendam esses requisitos. O entrelaçamento entre requisitos é demonstrado com um “X” que demonstra o

encontro entre um requisito funcional afetado por um não funcional. O Quadro 10 demonstra o mapeamento dos requisitos em elementos de projeto no presente estudo de caso.

Quadro 10. Mapeamento dos Requisitos em Elementos de Projeto para o Sistema de Ventilação do Ventilador Pulmonar

		Requisitos Não Funcionais								
		RNF-1	**RNF-2**	**RNF-3**	**RNF-4**	**RNF-5**	**RNF-6**	**RNF-7**	**RNF-8**	**Classes responsáveis pelo tratamento do RF**
Requisitos Funcionais	**RF-1**									*BreathControl* *Parameter*
	RF-2		**X**		**X**	**X**	**X**		**X**	*BreathControl* *Actuator* *PatientMode*
	RF-3	**X**		**X**		**X**	**X**		**X**	*Parameter* *Controlled_Parameter* *Sensor*
	RF-4							**X**		*Alarm_Manager* *Alarm* *AlarmView*
	Aspectos responsáveis pelo tratamento do RNF	*Periodic Timing* *Scheduling Support*	*Periodic Timing* *Scheduling Support*	*Periodic Timing* *Scheduling Support*	*Timing Atributes*	*Timing Atributes*	*Data Freshness*	*Timing Atributes*	*Timing Atributes* *Scheduling Support*	

Os aspectos *PeriodicTiming*, *TimingAtributes* e *SchedulingSupport* fazem parte do pacote *Timing* mencionado anteriormente. O aspecto *PeriodicTiming* adiciona o mecanismo de ativação periódica em objetos ativos. Para isto é preciso introduzir um novo atributo temporal (período), além do mecanismo de controle de execução periódica. O aspecto *SchedulingSupport* insere o mecanismo de escalonamento no sistema que dá suporte a execução de objetos ativos. Além disto, este aspecto é responsável pela inclusão de objetos ativos à lista de escalonamento e, ainda, por realizar o teste de verificação da escalonabilidade da lista. O aspecto *TimingAtributes* introduz os atributos temporais aos objetos ativos do sistema. Ele também é responsável pela inicialização dos atributos que introduz (FREITAS, 2007).

O aspecto *DataFreshness* faz parte do pacote *Precision* e associa *timestamps* a dados, a fim de verificar a sua validade antes de utilizá-los. Depois que um dado com validade controlada for escrito, seu *timestamp* deve ser atualizado. De forma análoga, a validade de um dado controlado deve ser verificada antes de se realizar uma leitura para utilização de seu valor. Caso a validade do dado tenha expirado, ações corretivas devem ser adotadas (FREITAS, 2007).

5.4. Projeto

Com base nas informações das etapas anteriores, constroem-se os diagramas que constituem o modelo do sistema do ventilador pulmonar. O primeiro a ser construído é o digrama de classes, apresentado na Figura 10. Nele, não foram apresentados os métodos e atributos das classes por uma questão de maior compreensão do diagrama. Na sua construção foram utilizados os estereótipos da RT-UML (*Real-Time UML*) *<<SAshedRes>>* e *<<SAresource>>*, que caracterizam o objeto como ativo e como recurso de acesso concorrente, respectivamente.

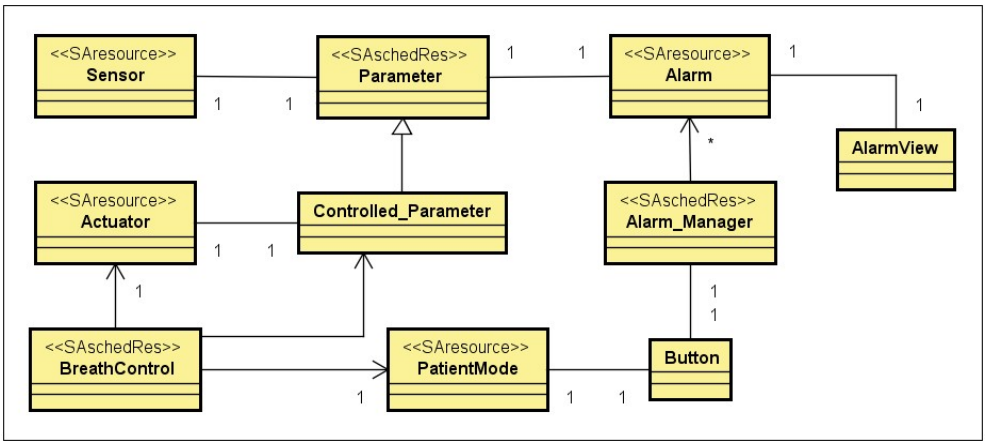

Figura 10. Diagrama de Classes

Em seguida, o diagrama ACOD (*Aspect Crosscutting Overview Diagram*) é construído (Figura 11). Neste diagrama são apresentadas somente as classes que possuem suas funcionalidades afetadas pelos requisitos não funcionais e os aspectos utilizados para tratar estes requisitos. O entrelaçamento entre o aspecto e a classe afetada é caracterizado pelo relacionamento navegável utilizando o estereótipo *<<crosscut>>*.

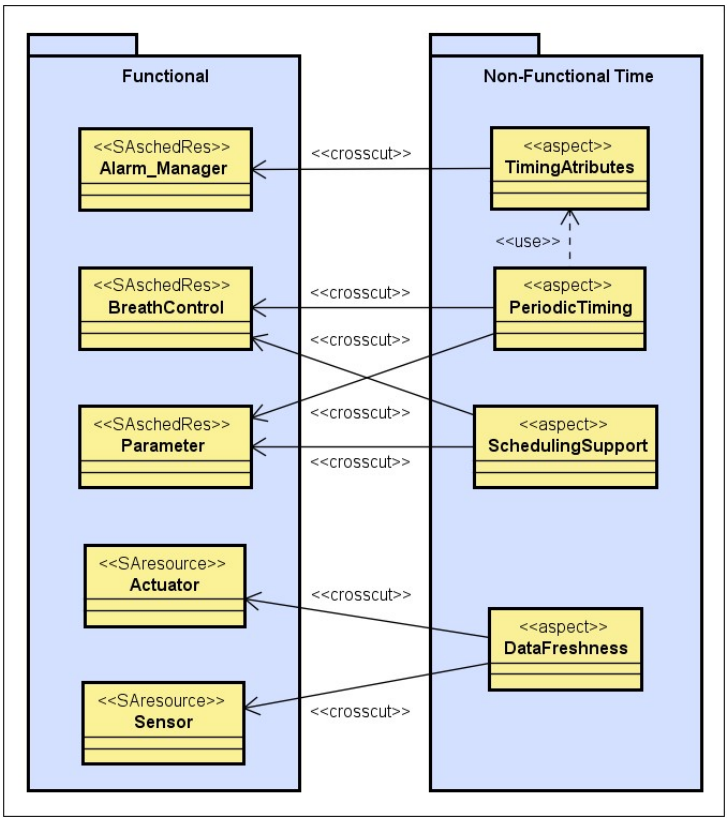

Figura 11. Diagrama ACOD

Com a utilização de aspectos ocorre a concentração do tratamento de requisitos não funcionais em um único elemento do projeto. Como pode ser visto na Figura 11, todos os elementos que caracterizam preocupação com requisitos temporais, como

limites de tempo para execução ou períodos, encontram-se especificados em um único elemento do projeto, dedicado apenas a este tipo de requisito não funcional. Evita-se com isto o espalhamento deste requisito não funcional por diversas entidades distintas do sistema. O tratamento não funcional afeta aquelas diversas entidades, porém a sua especificação encontra-se concentrada em uma entidade dedicada.

6. Avaliação dos Resultados

Com a finalidade de aferir o impacto da utilização da metodologia desenvolvida, as métricas de avaliação mencionadas na Seção 2.4 foram aplicadas ao estudo de caso realizado na Seção 5. Em seguida, os resultados obtidos por meio da avaliação foram comparados aos resultados da avaliação do modelo do mesmo sistema desenvolvido por Douglass (1998), porém, utilizando o paradigma orientado a objetos.

A capacidade de reuso e facilidade de manutenção são as qualidades analisadas no estudo de caso. Para que isso seja possível, o modelo de qualidade utilizado é o proposto por Sant'Anna *et. al* (2003). Este modelo de qualidade é baseado em um conjunto de métricas que se adaptam ao contexto de orientação a aspectos, motivo pelo qual foi escolhido para tal fim. As métricas referentes à implementação do sistema não foram utilizadas na avaliação do estudo de caso, pois esta fase não é alvo do presente estudo. Foram eliminadas da avaliação as métricas Difusão de Conceitos por Linhas de Código (DCLC), Linhas de Código (LC) e Peso de Operações por Componente (POC).

Os resultados da aplicação do conjunto de métricas relacionadas à separação de conceitos (preocupações ou interesses) do sistema do ventilador pulmonar são apresentados no Quadro 11.

Quadro 11. Resultado das Métricas de Separação de Conceitos aplicadas ao Projeto do Ventilador Pulmonar

Preocupação	**Tempo**	
Métricas	**DCC**	**DCO**
Projeto OO	7	14
Projeto OA	3	5
Resultado	**+57%**	**+64%**

No Quadro 11 pode-se notar os resultados das métricas Difusão de Conceito por Componentes (DCC) e Difusão de Conceitos por Operações (DCO) que envolvem a preocupação de tempo para os projetos orientados a objetos e a aspectos. Nota-se que ocorre uma melhora na separação das preocupações no projeto orientado a aspectos, pelo fato deste reunir em componentes específicos (aspectos) e que se encontravam espalhados na orientação a objetos (classes). Ocorreu, desta maneira, a redução no valor das métricas do Projeto OA com relação ao Projeto OO. A métrica DCC teve uma melhora de 57% e a métrica DCO teve uma melhora de 64%. Em um sistema com mais objetos ativos (com estereótipo *SAschedRes*), o resultado poderia ser ainda maior, visto que existiriam mais preocupações espalhadas pelos componente do sistema no projeto OA.

O Quadro 12 apresenta o resultado das métricas de acoplamento, coesão e tamanho aplicadas aos projetos OO e OA do ventilador pulmonar.

Quadro 12. Resultado das Métricas de Acoplamento, Coesão e Tamanho aplicadas ao projeto do Ventilador Pulmonar

Atributos Internos	Acoplamento		Coesão	Tamanho	
Métricas	**AEC**	**PAH**	**FCO**	**TV**	**NA**
Projeto OO	34	1	62	19	42
Projeto OA	30	1	46	23	34
Resultado	**+12%**	**0**	**+25%**	**-21%**	**+19%**

Pode-se observar, segundo os resultados, que houve uma pequena melhora com relação ao acoplamento. Na métrica Acoplamento Entre Componentes (AEC) a melhora para o AO ficou em torno de 12%. Na métrica Profundidade da Árvore de Herança (PAH) os valores foram os mesmos. Com relação à coesão, a métrica Falta de Coesão nas Operações (FCO) teve uma melhora de 25%. Quanto à métrica Tamanho do Vocabulário (TV), o projeto OA apresentou um valor maior que o projeto OO, representando, dessa forma, uma desvantagem de 21% para o OA. Esse dado pode ser explicado pelo fato do sistema não possuir restrições de distribuição e embarcados, e, ainda, ser de uma pequena dimensão. A métrica Número de Atributos (NA) teve uma melhora de 19% devido à concentração de atributos que no projeto OO estavam espalhados nas classes e no projeto OA encontram-se nos aspectos. É importante observar que o atributo de tamanho não afeta o fator flexibilidade, do qual é um dos responsáveis por aferir qualidade ao projeto (SANT´ANNA *et. al*, 2003), conforme pode ser observado na Figura 3. Dessa forma, o resultado final não é diretamente afetado pelo resultado negativo ocorrido na métrica Tamanho do Vocabulário (TV) do projeto OA.

Se considerarmos pesos iguais para os quatro atributos, tem-se uma melhora de aproximadamente 24% para o fator de compreensão e aproximadamente 32% para o fator flexibilidade. Como visto anteriormente, estes fatores estão ligados às características de reuso e de manutenibilidade. Pode-se, dessa forma, dizer que o uso de orientação a aspectos melhorou em torno de 28% estas qualidades no projeto do ventilador pulmonar.

7. Considerações Finais

O presente estudo teve como propósito a identificação e especificação de requisitos não funcionais utilizando o paradigma de orientação a aspectos em sistemas ciberfísicos. Para isso, utilizou-se a metodologia RT-FRIDA, adaptada por Freitas (2007) para Sistemas de Tempo-Real Embarcados Distribuídos, a partir da metodologia FRIDA (BERTAGNOLLI, 2004).

A utilização da referida metodologia no projeto do ventilador pulmonar foi de suma importância, principalmente, por se preocupar com o tratamento dos requisitos

não funcionais desde o seu início, trazendo resultados positivos quanto à facilidade de compreensão, reutilização de componentes e manutenção.

Buscou-se, neste trabalho, demonstrar os benefícios que o uso de aspectos traz ao desenvolvimento de um projeto quando comparado à orientação a objetos. Buscou-se, também, comparar tais paradigmas com o desenvolvimento de um estudo de caso, apresentando seus resultados.

O trabalho teve como principal contribuição demonstrar as etapas percorridas durante todo o processo de desenvolvimento do projeto e, especialmente, a sua importância no desenvolvimento de sistemas orientados a aspectos, corroborado com o estudo de caso. Porém, algo a mais pode ser feito, no âmbito de trabalhos futuros, como a implementação do sistema e um maior detalhamento do entrelaçamento que ocorre entre as classes que são afetadas por requisitos não funcionais e os aspectos usados no tratamento destes requisitos.

8. Referências

Araújo, J., Moreira, A., Brito, I., Rashid, A. (2002). Aspect-Oriented Requirements with UML. In *Workshop on Aspect-Oriented Modeling with UML*.

Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., Woodger, M. (1969). Revised Report on the Algorithmic Language ALGOL 60. *Programming Systems and Languages*, p. 71 – 117.

Bertagnolli, S. d. C. (2004). FRIDA: Um Método para Elicitação e Modelagem de RNFs. Programa de pós-graduação em computação, Dissertação de Doutorado, UFRGS, Porto Alegre.

Brito, I. S., Moreira, A. (2003a). Advanced Separation of Concerns for Requirements Engineering. In *JISBD*, p. 47 – 56.

Brito, I. S., Moreira, A. (2003b). Towards a Composition Process for Aspect-Oriented Requirements. In *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*.

Brito, I. S., Moreira, A. (2004). Integrating the NFR Framework in a RE Model. In *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*, Lancaster, UK.

Button, V. L. S. N. (2002). Equipamentos médico-hospitalares e o gerenciamento da manutenção. In *Ministério da Saúde, Secretaria de Gestão de Investimentos em Saúde*, Projeto REFORSUS. Brasília.

Chung, L., Nixon, B. A., Yu, E., Mylopoulos, J. (2000). Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers.

Douglass, B. P. (1998). Designing Real-Time Systems with UML-Part II. In *Embedded Systems Programming*, v. 11, p. 42 – 65.

Fornazier, C., Trindade, E., Holsbah, L. R., Barbieri, D. X., Perlato, M. T., Glowacki, L. A., Vicente, M. G., Pereira, A. L., Silva, J. E. L. d. (2011). Abordagem de Vigilância Sanitária de Produtos para Saúde Comercializados no Brasil: Ventilador Pulmonar.

In *BIT – Boletim Informativo de Tecnovigilância*, Brasília-DF, nº 03, jul/ago/set 2011.

Freitas, E. P. (2007). Metodologia Orientada a Aspectos para a Especificação de Sistemas Tempo-Real Embarcados Distribuídos. Programa de Pós-graduação em Computação, Dissertação de Mestrado, UFRGS, Porto Alegre.

Gill, H. (2011). Cyber-Physical Systems: Beyond ES, SNs, and SCADA. In *Trusted Computing in Embedded Systems (TCES) Workshop*.

Khaitan, S. K., McCalley, J. D. (2014). Design Techniques and Applications of Cyberphysical Systems: A Survey. In *IEEE Systems Journal*, v. 9, n. 2, p. 350 – 365.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J., Irwin, J. (1997). Aspect-Oriented Programming. In *ECOOP*, p. 220 – 242.

Lee, E. A. (2015). The Past, Present and Future of Cyber-Physical Systems: A Focus on Models. In *Sensors*, v. 15, n. 3, p. 4837 – 4869.

Lee, E. A., Seshia, S. A. (2014). Introduction to Embedded Systems - A Cyber-Physical Systems Approach. LeeSeshia.org, 1.5 edition. Disponível em: <http://leeseshia.org/>. Acesso em 10, abril, 2015.

Leite, J., Oliveira, A. (1995). A Client Oriented Requirements Baseline. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, p. 108 – 115, IEEE.

Perel, A., Stock, M. C. (1994). Manual de Mecanismos de Suporte Ventilatório. Rio de Janeiro, Medsi.

Rashid, A., Sawyer, P., Moreira, A. M. D., Araújo, J. (2002). Early aspects: a model for aspect-oriented requirements engineering. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, p. 199 – 202. IEEE.

Sanislav, T., Miclea, L. (2012). Cyber-Physical Systems - Concept, Challenges and Research Areas. In *Journal of Control Engineering and Applied Informatics*, v. 14, n. 2, p. 28 – 33.

Sant'Anna, C., Garcia, A., Chavez, C., Lucena, C. e v. von Staa, A. (2003). On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework. In *Proceedings XVII Brazilian Symposium on Software Engineering*, p. 19 – 34.

Shi, J., Wan, J., Yan, H., Suo, H. (2011). A Survey of Cyber-Physical Systems. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, p. 1 – 6. IEEE.

Sommerville, I. (2007). Software Engineering. Pearson Addison-Wesley, São Paulo, 8 edition.

Turrin, B. B. (2011). *Projeto e Desenvolvimento de um Sistema de Controle para um Dispositivo de Ventilação Mecânica Pulmonar*. Dissertação de Mestrado. Departamento de Engenharia de Telecomunicações e Controle. Escola Politécnica da Universidade de São Paulo, São Paulo.

Zhang, L. (2012). Aspect-Oriented Development Method for Non-Functional Characteristics of Cyber Physical Systems Based on MDA Approach. In *Journal of Software*, v. 7, n. 3, p. 608 – 619.

ANEXO 1 - LÉXICO PARA STrED

<RNF_genérico> ::= <tempo> | <desempenho> | <distribuição> | <embarcados>
<tempo> ::= <temporização> | <precisão>
<desempenho> ::= <tempo_de_resposta> | <vazão>
<distribuição> ::= <alocação_de_tarefas> | <estações_participantes> | <comunicação> | <sincronização>
<embarcados> ::= <área> | <consumo_potência> | <energia_total> | <memória>

Figura 1.1. Entrada do Léxico [Freitas 2007]

<tempo> ::= <temporização> | <precisão>
<temporização> ::= <deadline> | <período> | <custo> | <wcet> | <instante_de_liberação> | <latência_de_ativação> | <início_fim>
<deadline> ::= a execução deve ser feita até <n> <unidade_tempo>
<período> ::= a cada <n> <unidade_tempo> | por <unidade_tempo> | periodicamente | ciclicamente
<custo> ::= consome <n> <unidade_tempo>
<wcet> ::= no pior caso o tempo de execução deve ser de <n> <unidade_tempo>
<instante_de_liberação> ::= a atividade deve estar pronta para executar em <n> <unidade_tempo>
<latência_de_ativação> ::= após a liberação a atividade deve ser iniciada em <n> <unidade_tempo>
<início_fim> ::= atividade inicia em <n> <unidade_tempo> | termina em <n> <unidade_tempo>
<precisão> ::= <jitter> | <retardo_admitido> | <rigidez> | <utilidade> | <resolução> | <exatidão>
<jitter> ::= variação menor que <n> <unidade_tempo>
<retardo_admitido> ::= tempo excedente admitido para a execução de uma atividade é <n> <unidade_tempo>
<rigidez> ::= hard | médium | soft
<utilidade> ::= válido até <n> <unidade_tempo> | válido por <n> <unidade_tempo>
<resolução> ::= registra até <unidade_tempo>
<exatidão> ::= desvio de <n> <unidade_tempo>
<unidade_tempo> ::= h | min | s | ms | μs | ns | hora | minuto | segundo | milissegundo | microssegundo | nanossegundo | dia | semana | mês | ano
<n> ::= <n> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figura 1.2. Léxico para o Contexto tempo [Freitas 2007]

<desempenho> ::= <tempo_de_resposta> | <vazão>

<tempo_de_resposta> ::= o tempo para o sistema retornar uma resposta final é de <n> <unidade_tempo>

<vazão> ::= chamadas por <unidade_tempo> | execuções por <unidade_tempo>

<unidade_tempo> ::= h | min | s | ms | μs | ns | hora | minuto | segundo | milissegundo | microssegundo | nanossegundo | dia | semana | mês | ano

<n> ::= <n> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figura 1.3. Léxico para o Contexto Desempenho [Freitas 2007]

<distribuição> ::= <alocação_de_tarefas> | <estações_participantes> | <comunicação> | <sincronização>

<alocação_de_tarefas> ::= <n> tarefas devem ser alocadas em determinada estação | determinada tarefa deve ser alocada em uma estação específica

<estações_participantes> ::= a estação suporta executar <n> tarefas simultaneamente | a estação deve operar com ocupação de <n>% de sua capacidade de processamento

<comunicação> ::= com garantia de entrega | sem garantia de entrega | deve usar <tecnologia_comunicação> | taxa de <n> <unidade_dados> por <unidade_tempo>

<sincronização> ::= <n> processo(s) pode(m) ler um dado ao mesmo tempo | um recurso pode ser compartilhado por <n> processo(s)

<tecnologia_comunicação> ::= CAN | TT-CAN | FTT-CAN | CAN-Aero | Ethernet | FieldBus | ProfiBus | IEEE802.11

<unidade_dados> ::= bit | byte | Kbit | Kbyte | Megabit | Megabyte | Gigabit | Gigabyte | Terabyte

<unidade_tempo> ::= h | min | s | ms | μs | ns | hora | minuto | segundo | milissegundo | microssegundo | nanossegundo | dia | semana | mês | ano

<n> ::= <n> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figura 1.4. Léxico para o Contexto Distribuição [Freitas 2007]

<embarcados> ::= <área> | <consumo_potência> | <energia_total> | <memória>
<área> ::= <n> de células lógicas | <n> <unidade_comprimento>²
<consumo_potência> ::= <n> Watts | <n> Joules por operação
<energia_total> ::= <n> Joules | <n> Joules por componente | <autonomia>
<autonomia> ::= <n> de operações | <n> de operações por <unidade_tempo> | <n> <unidade_comprimento> percorrida | <n> <unidade_tempo>
<memória> ::= <n> <unidade_dados> de armazenamento | <n> <unidade_dados> de memória física | <n> <unidade_dados> de memória volátil | ocupa <n> <unidade_dados>
<unidade_comprimento> ::= km | m | dm | cm | mm | quilometro | metro | decímetro | centímetro | milímetro
<unidade_tempo> ::= h | min | s | ms | μs | ns | hora | minuto | segundo | milissegundo | microssegundo | nanossegundo | dia | semana | mês | ano
<unidade_dados> ::= bit | byte | Kbit | Kbyte | Megabit | Megabyte | Gigabit | Gigabyte | Terabyte
<n> ::= <n> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figura 1.5. Léxico para o Contexto Embarcados [Freitas 2007]