

UNIVERSIDADE CATÓLICA DE BRASÍLIA

PROGRAMA DE PÓS-GRADUAÇÃO LATO SENSU

ENGENHARIA DE SOFTWARE

**APLICAÇÃO DA ABORDAGEM GQM PARA A DEFINIÇÃO DE UM
PROCESSO DE ENGENHARIA DE REQUISITOS DE SOFTWARE
EMBARCADO**

Autores: Daniele Erica Damke

Patrícia Freitas de Moraes

Orientadora: Prof^a MSc. Claudia de Oliveira Melo

BRASÍLIA

2008

DANIELE ERICA DAMKE
PATRÍCIA FREITAS DE MORAES

Trabalho apresentado ao Programa de Pós-Graduação Latu Sensu em Engenharia de Software da Universidade Católica de Brasília, como requisito para obtenção do Título de Especialista em Engenharia de Software.

Orientadora: MSc. Claudia de Oliveira Melo.

BRASÍLIA

2008

Este trabalho é dedicado a todas as pessoas que nos acompanharam na elaboração deste trabalho contribuindo para seu êxito.

AGRADECIMENTOS

Agradecemos às empresas participantes da pesquisa e ao Prof Dr. João Alberto Fabro pela colaboração e apoio na pesquisa.

À nossa orientadora Prof^a MSc. Claudia de Oliveira Melo e a todos os professores que nos orientaram nessa jornada.

Aos nossos amigos que incentivaram nossa caminhada.

Aos nossos companheiros pela compreensão dos momentos de ausência durante a elaboração do trabalho.

A nossa família pelo incentivo e pelo crédito depositado em nós.

A todos que de alguma forma colaboram com as pesquisas e com o crescimento de um mercado de software de maior qualidade.

A Deus por sempre nos direcionar e iluminar, permitindo a conclusão deste projeto.

Tudo dá certo no final.

Se não deu certo é porque ainda não chegou o fim.

RESUMO

O mercado de software embarcado vem crescendo exponencialmente nos últimos anos. Apesar disso, o desenvolvimento deste tipo de software ainda é uma área que merece atenção dos pesquisadores devido a sua complexidade e a falta de técnicas e ferramentas específicas para esse domínio de sistemas.

Aliada a todas as dificuldades do desenvolvimento de sistemas embarcados o mercado nacional de desenvolvimento desse segmento de software é composto em sua maioria por Micro e Pequenas Empresas (MPEs). O que de certa forma agrava o problema devido fato desse tipo de empresas possuir pouca informalidade em seus processos.

Este trabalho realizou uma avaliação do processo de desenvolvimento de micro e pequenas empresas desenvolvedoras de software embarcado com o objetivo de definir um processo de engenharia de requisitos para sistemas embarcados.

Palavras-chave: Sistemas Embarcados, Micro e Pequenas Empresas, Processo de Requisitos.

LISTA DE ILUSTRAÇÕES

Figura 1 - Onde estão os processadores? (Tennenhouse 2000)	19
Figura 2 – Modelo de Processo ISO/IEC 12207 (Calsalvara, et al. 2000).....	31
Figura 3 - Inter-relação entre os elementos de um modelo de processo de software (Weber 2002)	33
Figura 4 - Gráfico das Baleias do RUP (Kruchten 2003).....	37
Figura 5 - Gráfico das Baleias do IPProcess (Barros, Bione, et al. 2005)	39
Figura 6 - Gráfico Y do Modelo UPES	41
Figura 7 - Causas de Retrabalho (Gastaldo 2003).....	43
Figura 8 – Classificação de Requisitos Não-Funcionais (Sommerville 1998).....	46
Figura 9 - Balanceamento da Equipe de ER (Hofmann 2001)	50
Figura 10 - Fluxo de Trabalho da Disciplina de Requisitos (RUP 2001).....	52
Figura 11 - Decomposição do processo de desenvolvimento de sistemas embarcados (Graaf 2003)	59
Figura 12 - Processo de Especificação de Requisitos (Lattermann 1997)	60
Figura 13 - Interação indireta entre o piloto e um caso de uso do TRCS (Nars 2002)	61
Figura 14 - Os principais construtores da técnica de caso de uso (Nars 2002).....	63
Figura 15 - Interação indireta entre o piloto e um caso de uso do TRCS (Nars 2002)	64
Figura 16 - Envolvidos do desenvolvimento de sistemas embarcados (Graaf 2003)	64
Figura 17 - Fases da Abordagem GQM (Berghout e Solligen 1999)	70
Figura 18 – Modelo de Avaliação GQM (Laboratory s.d.)	74
Figura 19 - Relação entre os Elementos do Modelo GQM.....	96
Figura 20 - Tela Inicial do ProReqSE.....	106
Figura 21 – Fluxo da Atividade Definir Visão	109
Figura 22 – Fluxo da Atividade Encontrar e Resumir Requisitos	113
Figura 23 – Fluxo da Atividade Detalhar Requisitos	115
Figura 24 – Fluxo da Atividade Revisar Requisitos.....	117

LISTA DE TABELAS

Tabela 1 - Componentes do SPEM	32
Tabela 2 - Comparativo das Características de ER em Processos de	42
Tabela 3 - Melhores Práticas da ER (Hofmann 2001).....	48
Tabela 4 – Definição das palavras-chave da linguagem de especificação PLanguage (Gastaldo 2003)	55
Tabela 5 – Questões de Definição dos Objetivos da Medição (Laboratory s.d.)	72
Tabela 6 - <i>Abstract Sheets</i> (Berghout e Solligen 1999)	73
Tabela 7 - Questões Iniciais da Fase de Planejamento	78
Tabela 8 - Equipe GQM.....	78
Tabela 9 – Modelo de Definição dos Objetivos da Medição (Laboratory s.d.)	80
Tabela 10 - Objetivo do Negócio	80
Tabela 11 - Pontos de Verificação.....	81
Tabela 12 - Contextualização da Empresa	83
Tabela 13 - Contextualização do Software	84
Tabela 14 - Contextualização da Equipe de Requisitos	85
Tabela 15 - Entendimento dos Requisitos Funcionais e Não-Funcionais.....	87
Tabela 16 – Identificação e Registro das Necessidades, Expectativas e Restrições do Cliente	89
Tabela 17 - Modo de Especificação dos Requisitos.....	91
Tabela 18 – Definição e Manutenção dos Requisitos Funcionais e Não – Funcionais.....	91
Tabela 19 - Rastreabilidade dos Requisitos.....	92
Tabela 20 – Análise dos Requisitos.....	93
Tabela 21 – Revisão dos Planos e Produtos de Trabalho	94
Tabela 22 - Identificação das Interfaces do Sistema.....	95
Tabela 23 - Índices dos Pontos de Verificação	99
Tabela 24 - Comparativo das Características de ER em Processos de	120

LISTA DE ABREVIações

CI	Circuitos Integrados
DBL	Desenvolvimento Baseado em Componentes
EPF	<i>Eclipse Process Framework Composer</i>
ER	Engenharia de Requisitos
EUA	Estados Unidos da América
GQM	<i>Goal /Question / Metric</i>
Intel Corp.	<i>Intel Corporation</i>
IpProcess	Processo de Desenvolvimento de Soft-Ip
ISO/IEC	<i>International Organization for Standardization / International Electrotechnical Commission</i>
JAD	<i>Joint Application Development/Design</i>
MDA	<i>Model Driven Architecture</i>
MPE	Micro e Pequena Empresa
MPS.BR	Programa de Melhoria de Software Brasileiro
NASA	<i>National Aeronautics and Space Administration</i>
OMG	<i>Object Management Group</i>
PLanguage	<i>Planning Language</i>
PU	Processo Unificado
RUP	<i>Rational Unified Process</i>
RUPSE	<i>Rational Unified Process for Sytem Engineering</i>
SE	Sistemas Embarcados
SPEM	<i>Software Process Engineering Metamodel</i>
UC	Caso de Uso
UML	<i>Unified Model Language</i>
UPES	<i>Unified Process for Embedded System</i>

SUMÁRIO

AGRADECIMENTOS	4
RESUMO.....	6
LISTA DE ILUSTRAÇÕES	7
LISTA DE TABELAS	8
LISTA DE TABELAS	8
LISTA DE ABREVIações	9
SUMÁRIO.....	10
1. INTRODUÇÃO	14
1.1 Motivação e Contexto	14
1.2 Objetivos.....	15
1.3 Resultados Esperados.....	16
1.4 Metodologia	16
1.5 Organização do Trabalho.....	17
2. SISTEMAS EMBARCADOS	18
2.1 Definição.....	19
2.1.1 Características de Sistemas Embarcados	20
2.2 Dificuldades na Evolução dos Sistemas Embarcados.....	23
2.2.1 Dificuldades Acadêmicas.....	23
2.2.2 Dificuldades no Projeto.....	24
2.3 Considerações Finais	27
3. PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE.....	29
3.1 Processo.....	29
3.2 Modelo de Processo de Software	30
3.2.1 Elementos do Modelo de Processo de Software.....	31
3.3 Modelagem de Processo	33
3.4 Modelos de Processos de Desenvolvimento de Software.....	35

3.4.1	Processo Unificado.....	35
3.4.2	Rational Unified Process.....	36
3.5	Modelos de Processo de Desenvolvimento de Software Embarcado.....	37
3.5.1	RUPSE	37
3.5.2	IPProcess	38
3.5.3	Processo Unificado para Sistemas Embarcados (UPES).....	40
3.6	Considerações Finais do Capítulo	41
4.	REQUISITOS	43
4.1	Conceitos Básicos	44
4.1.1	Requisitos.....	44
4.1.2	Requisitos Funcionais x Requisitos Não-Funcionais.....	45
4.1.3	Engenharia de Requisitos.....	46
4.1.4	Processo de ER Tradicional	50
4.1.5	Gerência de Requisitos	52
4.1.6	Envolvidos no Processo de Engenharia de Requisitos	53
4.1.7	Revisões de Requisitos	53
4.2	Requisitos em Sistemas Embarcados.....	53
4.2.1	Requisitos Não Funcionais	53
4.2.2	Requisitos de Hardware x Requisitos de Software.....	56
4.2.3	Invenção de Requisitos	56
4.3	Processo de ER para Sistemas Embarcados.....	57
4.3.1	Contexto das Empresas Desenvolvedoras de SE.....	57
4.3.2	Processos de Desenvolvimento de SE Sob o Ponto de Vista da ER	58
4.3.3	Metodologias para Especificação dos Requisitos em SE.....	61
4.3.4	Envolvidos no Processo de ER para SE	64
4.3.5	Modelos de Documentos para Especificação dos Requisitos em SE.....	65
4.3.6	Gerenciamento de Requisitos em SE	66
4.3.7	Outros Aspectos Importantes no Processo de ER para SE	66

4.4 Considerações Finais	67
5. ABORDAGEM GQM.....	68
5.1 O Método Goal Question Metric (GQM)	68
5.1.1 Planejamento.....	70
5.1.2 Definição	71
5.1.3 Coleta	74
5.1.4 Interpretação	75
5.1.5 Captura de Experiências.....	76
5.2 Considerações Finais do Capítulo	76
6. AVALIAÇÃO DO PROCESSO DE ENGENHARIA DE REQUISITOS PARA O DESENVOLVIMENTO DE SISTEMAS EMBARCADOS	77
6.1 Aplicação do GQM.....	77
6.1.1 Planejamento.....	78
6.1.2 Definição	79
6.1.3 Elaboração das Questões e Métricas	82
6.1.4 Coleta	97
6.1.5 Interpretação	97
6.1.6 Captura de Experiências.....	102
6.2 Considerações Finais	102
7. DEFINIÇÃO DO PROCESSO DE ENGENHARIA DE REQUISITOS PARA O DESENVOLVIMENTO DE SISTEMAS EMBARCADOS	104
7.1 Processo de Engenharia de Requisitos para Sistemas Embarcados – ProReqSE.....	104
7.1.1 Papéis	107
7.1.2 Atividades.....	108
7.1.3 Produtos de Trabalho	117
7.1.4 Diretrizes	119
7.1.5 Listas de Verificação.....	119
7.1.6 Conceitos	120
7.2 Comparação do ProReqSE com Outros Processos	120

7.3 Considerações Finais	121
8. CONSIDERAÇÕES FINAIS DO TRABALHO	122
8.1 Conclusão.....	122
8.2 Trabalhos Futuros.....	123
REFERÊNCIAS	124
APÊNDICE A – RESPOSTA DO QUESTIONÁRIO DE AVALIAÇÃO DOS PROCESSOS DE DESENVOLVIMENTO DE SISTEMAS EMBARCADOS	128
APÊNDICE B – LISTA DE VERIFICAÇÃO I	134
APÊNDICE C – LISTA DE VERIFICAÇÃO II	138
APÊNDICE D – LISTA DE VERIFICAÇÃO III	142
APÊNDICE E – LISTA DE PRIORIZAÇÃO.....	146
APÊNDICE F – DOCUMENTO DE REFERÊNCIA CRUZADA	148
APÊNDICE G – DOCUMENTO DE ESPECIFICAÇÃO SUPLEMENTAR	152

1. INTRODUÇÃO

1.1 Motivação e Contexto

Aplicações de software embarcado estão cada vez mais presentes em nosso cotidiano. Em geral são sistemas extremamente críticos para o sucesso do negócio e muitas vezes para a segurança e bem-estar humanos, como as aplicações de medicina, telecomunicações e aviação. O projeto deste tipo de sistemas é complexo, pois apresenta restrições severas em relação às funcionalidades e implementação, dentre elas:

- a reação a eventos externos em tempo real;
- adaptação a limites de tamanho e peso;
- gerenciamento de consumo de potência sem perda de desempenho;
- baixa disponibilidade de memória;
- satisfação a requisitos de confiabilidade e segurança e ;
- adequação a restrições de orçamento.

Além disso, o projeto de sistemas embarcados é bastante caro para as empresas por envolver equipes especializadas e multidisciplinares (hardware digital, hardware analógico, software, teste), e abordar conceitos pouco estudados pela computação de propósitos gerais.

A diminuição dos custos de produção, o avanço da fabricação de hardware e software e a necessidade crescente do mercado possibilitaram o desenvolvimento de complexas aplicações de software embarcado *ad hoc*. Por outro lado, o aumento de complexidade e criticidade destas aplicações refletiram-se no processo de desenvolvimento destes sistemas, sendo necessária uma maior ênfase na verificação e validação de suas atividades.

A evolução das ferramentas específicas para desenvolvimento de softwares embarcados auxilia o processo, entretanto, as estratégias de desenvolvimento adotadas nem sempre resultam em uma solução de qualidade satisfatória. Estudos com mais de 450 desenvolvedores de sistemas embarcados identificaram alguns fatores que contribuem para o aumento do número de projetos fracassados (atrasados ou cancelados): mudança na especificação, especificação incompleta, complexidade da aplicação e quantidade insuficiente de desenvolvedores (Carro 2002). A avaliação e identificação das deficiências de estratégias que auxiliam o desenvolvimento de requisitos não-funcionais como previsibilidade, confiabilidade, desempenho, tamanho reduzido, criticidade e suporte a tempo real tornam-se objeto de estudo para melhoria da qualidade das soluções.

Outro ponto que deve ser ressaltado é o crescimento e sucesso da indústria brasileira de desenvolvimento de software. Para que o setor torne-se realmente competitivo nacional e internacionalmente é necessário que haja um incentivo as empresas para a que os produtos e serviços oferecidos atendam aos padrões de qualidade internacionalmente exigidos (Softex s.d.). No entanto, o setor de desenvolvimento de software brasileiro é composto em sua grande maioria por micro e pequenas empresas (MPEs) (MCT 2002). Devido a restrições de custos e investimentos a principal característica dessa parcela do mercado brasileiro é a informalidade das próprias empresas e de seus processos (J. C. Hauck 2004).

Assim, é de vital importância a pesquisa e o incentivo para que esse tipo de empresas possa melhorar seus processos com o intuito de adequar-se às normas nacionais e internacionais de desenvolvimento de software e melhorar a qualidade dos produtos e serviços oferecidos no mercado (Softex s.d.). Para que isso aconteça, é necessário que os processos da empresa estejam corretamente definidos, documentados e institucionalizados com cada uma das atividades da organização representadas em um modelo (J. C. Hauck 2004). Segundo Hauck, A institucionalização do processo de desenvolvimento proporciona às organizações um desenvolvimento estruturado de seus projetos de forma que diversas pessoas possam desempenhar mais de uma função e o resultado final do produto atinja os mesmo níveis de qualidade independente de quem o desenvolveu.

1.2 Objetivos

Este projeto baseia-se em estudos das áreas de sistemas embarcados, processos de desenvolvimento e de engenharia de requisitos para sistemas embarcados e GQM

Neste contexto, esta proposta de pesquisa tem como objetivo definir e aplicar uma avaliação do processo de engenharia de requisitos de MPEs desenvolvedoras de Sistemas Embarcados (SE). Com base nessa avaliação e nas características específicas deste tipo de software, o trabalho pretende sugerir um processo de engenharia de requisitos mais efetivo e aderente à realidade das MPEs desenvolvedoras de software embarcado. Logo, o desenvolvimento de cada atividade é caracterizado por meio dos seguintes objetivos específicos:

- Elaborar revisão bibliográfica: consiste na pesquisa, seleção e estruturação de informações para construção do referencial teórico da monografia.
- Definir o Programa de Medições (GQM): planejar (definir a equipe GQM, escolher a área de melhoria e preparar os envolvidos), definir (documentar os objetivos, questões e métricas), coletar os dados e interpretá-los em relação às métricas definidas, gerando os resultados da medição.

- Propor e documentar ajustes em um processo de engenharia de requisitos: tomando como base na disciplina de Requisitos do OpenUP, os resultados do GQM e as definições do programa de melhoria de software brasileiro (MPS.Br).

Para a definição da avaliação foi selecionada a metodologia *Goal/Question/Metrics* (GQM) (Basili e Rombach 1994), que tem se apresentado um mecanismo eficiente no planejamento e definição de objetivos de medição e avaliação. Utilizada em diversos objetos de pesquisa na área de Engenharia de Software e adotada como uma importante ferramenta de avaliação de qualidade de software, a principal característica de GQM é a adaptabilidade para avaliação de qualquer tipo de problema. Logo, o método será aplicado nas MPEs participantes da avaliação.

1.3 Resultados Esperados

São esperado os seguintes resultados do trabalho:

- Identificar as principais limitações dos atuais processos de engenharia de requisitos de software embarcado com base na aplicação do método GQM em MPEs nacionais.
- Proposição de um processo de engenharia de requisitos que visa à diminuição da complexidade e dos custos, além do aumento da efetividade no desenvolvimento de sistemas embarcados, com base na pesquisa realizada. Este processo levará em consideração as características próprias deste tipo de sistema como:
 - Resposta em tempo real, resposta a eventos síncronos, tamanho e custo reduzidos, segurança e confiabilidade, ambientes inóspitos, previsibilidade, portabilidade, limites de tamanho e peso, gerenciamento de consumo de potência sem perda de desempenho, baixa disponibilidade de memória.

1.4 Metodologia

Este projeto é classificado como pesquisa aplicada e qualitativa por gerar conhecimentos para aplicação prática dirigida à solução de problemas do desenvolvimento de um tipo específico de software (Oliveira s.d.). A pesquisa será avaliada em um conjunto exclusivo de empresas cujos dados serão analisados indutivamente pelos pesquisadores e de acordo com a abordagem de estudo. Quanto aos fins, esse trabalho é uma pesquisa metodológica por elaborar um instrumento de captação da realidade, definindo formas para atingir o objetivo de estudo. Quanto aos meios, essa pesquisa será bibliográfica - baseada em livros, artigos científicos, redes eletrônicas e documental.

1.5 Organização do Trabalho

Este trabalho é composto de sete capítulos organizados da seguinte forma:

- O capítulo 2 descreve conceitos relacionados a sistemas embarcados, além da exposição de suas características e dos problemas de seu desenvolvimento.
- O capítulo 3 discorre sobre a elaboração de processos de desenvolvimento de software e sobre os elementos que os compõem. São descritos ainda os principais processos de desenvolvimento de software tradicionais e específicos para sistemas embarcados.
- O capítulo 4 apresenta os principais processos de engenharia de requisitos tradicionais e específicos para sistemas embarcados.
- O capítulo 5 descreve o método GQM, suas fases e os produtos gerados.
- O capítulo 6 apresenta todo o processo de elaboração e aplicação do programa de medições.
- O capítulo 7 descreve a adaptação do processo de engenharia de requisitos do processo unificado as características dos sistemas embarcados.
- O capítulo 8 apresenta as conclusões desta monografia, destacando as contribuições desta pesquisa e os trabalhos futuros.

2. SISTEMAS EMBARCADOS

A tecnologia da computação tem se tornado onipresente no dia-a-dia da população. Softwares embarcados hoje controlam a comunicação, os transportes, os sistemas médicos, entre outros (Henzinger 2007). Embora apenas recentemente os sistemas embarcados tenham se mostrado extremamente populares no cotidiano das pessoas, esta categoria de sistemas surgiu e foi reconhecida nos Estados Unidos no início da década de 60 em um computador-guia do projeto Apollo – denominação do conjunto de missões espaciais coordenadas pela NASA (*National Aeronautics and Space Administration*) – com o objetivo de levar o primeiro homem à lua (Wolf 2001).

Com a evolução dos microprocessadores e a diminuição de seus custos de produção, os sistemas embarcados tornaram-se universais, pois estão presentes em muitos dispositivos, desde os mais simples, como eletrodomésticos, até dispositivos complexos como os de naves espaciais. Entretanto, a maioria desses sistemas não é percebida pelas pessoas, pois como o próprio nome insinua, estão embutidos dentro de outros sistemas utilizados no dia-a-dia (Barreto 2006).

Em 2000 Tennenhouse (Tennenhouse 2000), na ocasião vice-presidente e diretor de pesquisas da *Intel Corporation* (Intel Corp) , publicou um artigo no qual mostrava que 80% de todos os processadores fabricados no mundo eram usados em sistemas embarcados (Figura 1). Os 20% restantes eram divididos em aplicações desktop (2%), aplicações robóticas (6%) e sistemas automotivos (12%). Neste contexto, não resta outra saída para as pesquisas em Ciência da Computação senão seguir a mesma direção dos processadores e investir uma boa parcela de seu capital intelectual neste novo espaço de pesquisa.



Figura 1 - Onde estão os processadores? (Tennenhouse 2000)

As aplicações mais comuns do mercado de processadores embarcados são telefones celulares, brinquedos, eletrodomésticos e outros equipamentos eletrônicos. Estes produtos são projetados sob restrições de recursos tais como energia consumida, utilização de memória, tamanho do circuito integrado, desempenho, resposta em tempo real, e, principalmente, custo. No custo deve ser incluído, além dos componentes de hardware, o desenvolvimento do hardware e de software aplicativo (Barbiero 2006).

Contudo, à medida que aumenta a quantidade e variedade de tais produtos, há um acréscimo em sua complexidade, devido à integração de mais componentes de software e hardware ao sistema. E, aliado a isso, ainda existe uma série de restrições como aquelas citadas no parágrafo anterior. A união destes aspectos torna o projeto de sistemas embarcados um desafio constante, visto que todas estas características devem ser levadas em consideração durante o desenvolvimento (Graaf 2003).

Este capítulo abordará, na seção 2.1, a definição e as características de sistemas embarcados. Já a seção 2.2 apresentará as dificuldades na evolução deste tipo de sistema. Por fim, a seção 2.3 descreve as considerações finais do capítulo.

2.1 Definição

Definir sistemas embarcados não é uma tarefa trivial. Segundo (Marwedel 2003), sistemas embarcados podem ser definidos como sistemas eletrônicos de processamento de informação embutidos em um produto de forma transparente para o usuário. Como já foi

descrito anteriormente, este tipo de sistema é composto de sistemas micro processados que não aparecem como funcionalidade final de um produto, ou seja, estão completamente encapsulados ou dedicados ao dispositivo ou sistema que ele controla, portanto, são invisíveis do ponto de vista do usuário (Wolf 2001).

Lee (E. Lee, Embedded Software 2002) ressalta, no entanto, que sistemas embarcados não se resumem a software em um pequeno dispositivo. Ao contrário da computação tradicional, sua principal característica não é a transformação de dados, mas preferencialmente a interação com o mundo físico. Um software cuja principal característica é a interação com o mundo real deve, por necessidade, adquirir algumas propriedades de tal mundo. Ou seja, ele gasta tempo, consome energia e não finaliza nunca (a menos que falhe). Além disso, são softwares que implementam um conjunto de tarefas pré-definidas, geralmente com requisitos específicos que são utilizados para acrescentar ou otimizar funcionalidades (Barreto 2006).

Em linhas gerais, os processos e componentes que compõem os sistemas embarcados são muito parecidos com a computação de propósito geral. No entanto, na computação tradicional, o objetivo principal é atender a funcionalidades com desempenho cada vez maior. Já na computação embarcada o interesse está voltado para interfaces com o ambiente, como sensores e atuadores, e algoritmos de controle, de modo a satisfazer severas restrições, como, requisitos temporais, de consumo de energia e memória, mobilidade, tamanho, peso, segurança, confiabilidade, tempo de entrega dentre outras (Barreto 2006).

2.1.1 Características de Sistemas Embarcados

A seguir são apresentadas algumas características comuns em sistemas embarcados, segundo (Marwedel 2003). Vale lembrar que os sistemas não necessariamente possuem todas estas características, mas devem ter muitas delas para serem considerados embarcados.

2.1.1.1 Acoplado ao ambiente físico

Freqüentemente, sistemas embarcados são conectados ao ambiente físico através de sensores que coletam informações sobre este espaço e atuadores que controlam este ambiente.

2.1.1.2 Confiável

Sistemas embarcados têm que ser confiáveis. Muitos sistemas embarcados são de segurança crítica, ou seja, são sistemas cujas falhas podem gerar conseqüências catastróficas, colocando em risco até mesmo vidas humanas. Sistemas de controle de

estações de energia nuclear são exemplos de sistemas de extrema segurança crítica que são, ao menos em parte, controlados por software. Confiabilidade é, entretanto, importante também em outros sistemas como carros, trens, aviões etc. Uma razão chave para ser de segurança crítica é que estes sistemas são diretamente conectados ao ambiente e têm um impacto imediato neste ambiente.

Um sistema considerado confiável deve abranger os seguintes aspectos (Pradhan 1996):

- Confiabilidade: probabilidade de que um sistema não vá falhar;
- Manutenibilidade: a probabilidade de que uma falha no sistema possa ser reparada dentro de uma certa janela de tempo;
- Disponibilidade: probabilidade de que o sistema esteja disponível. Os dois aspectos anteriores (confiabilidade e manutenibilidade) devem ser altos para garantir a alta disponibilidade;
- Segurança no funcionamento (*safety*): probabilidade de que uma falha no sistema não cause nenhum dano;
- Segurança no uso (*security*): propriedade de que dados confidenciais continuem confidenciais e que comunicações autênticas estejam garantidas.

2.1.1.3 Eficiente

Sistemas embarcados devem ser eficientes. Para garantir esta eficiência, algumas características devem ser levadas em consideração:

- Energia: muitos sistemas embarcados são móveis, ou seja, obtêm energia de baterias. Se por um lado os requisitos computacionais estão evoluindo rapidamente (especialmente para aplicações multimídia) e os usuários estão esperando que suas baterias durem cada vez mais. Por outro, a tecnologia das baterias não tem acompanhado a mesma velocidade de evolução. Sendo assim, a energia elétrica disponível para os sistemas embarcados deve ser usada com muita eficiência;
- Tamanho do código: todo o código que rodará em um sistema embarcado deve ser armazenado com o sistema. Normalmente, não há discos rígidos onde o código pode ser armazenado. Portanto, o tamanho do código deve ser o mínimo possível;
- Eficiência em tempo de execução: o mínimo de recursos deve ser usado para implementar as funcionalidades requeridas. A fim de reduzir o consumo de

energia, frequência de *clock* e voltagens é preciso ser o menor possível. Ou seja, apenas os componentes necessários devem estar presentes;

- Peso: todos os sistemas portáteis devem ter baixo peso;
- Custo: para a produção de sistemas em massa, principalmente os eletrônicos destinados ao consumo, competitividade no mercado é uma característica essencial.

2.1.1.4 *Dedicado à aplicação*

Estes sistemas são dedicados a uma determinada aplicação. Por exemplo, um processador que rode um software de controle de carro, sempre rodará este software. Não deve haver a possibilidade de executar um jogo ou um programa de planilhas no mesmo processador. Isto não é indicado devido a duas razões: aplicações adicionais diminuiriam a segurança do sistema e executá-las só seria viável caso os recursos (como memória) estivessem sem uso.

2.1.1.5 *Interface dedicada*

A maioria dos sistemas embarcados não usa teclados, mouses e monitores grandes de computadores para sua interface com o usuário. Ao invés disso, existem interfaces com o usuário dedicadas que consistem de botões de pressão, volantes, pedais, etc. Ademais, muitas das atividades executadas por estes sistemas são autônomas e grande parte de suas interações são com outros sistemas, ao invés de humanos. Por isso, o usuário dificilmente nota que o processamento de informações é envolvido. Por este motivo, a nova era da computação também foi caracterizada como *disappearing computer*.

2.1.1.6 *Restrição de Tempo Real*

Muitos sistemas embarcados devem suportar restrições de tempo real. Neste caso, não completar um processamento dentro da janela de tempo esperada pode resultar em uma perda significativa na qualidade provida pelo sistema (por exemplo, se a qualidade de áudio ou vídeo for afetada) ou em uma ameaça ao usuário (por exemplo, se carros, trens ou aviões não operarem da maneira prevista).

2.1.1.7 *Híbrido*

Muitos sistemas embarcados são sistemas híbridos no sentido de incluírem partes analógicas e digitais. As partes analógicas usam valores contínuos de sinal em tempos contínuos, ao passo que partes digitais usam valores discretos de sinais em tempos discretos.

2.1.1.8 Reativo

Tipicamente, sistemas embarcados são sistemas reativos, ou seja, estão em interação contínua com seu ambiente e executam em um ritmo determinado por este ambiente (Bergé, 1995, apud Marwedel, 2003, p.4). Os sistemas reativos são aqueles que permanecem em um determinado estado, esperando uma entrada. Para cada entrada, eles efetuam alguns processamentos e geram uma saída e um novo estado. Portanto, autômatos são modelos muito bons para tais sistemas. Funções matemáticas, as quais descrevem os problemas resolvidos pela maioria dos algoritmos, podem ser um modelo inapropriado.

2.2 Dificuldades na Evolução dos Sistemas Embarcados

Ainda que as oportunidades pareçam ilimitadas, a evolução dos softwares embarcados não tem conseguido acompanhar o potencial oferecido pelas tecnologias emergentes de hardware e de comunicação (Henzinger 2007). Nas subseções seguintes são apresentadas algumas das dificuldades encontradas no estudo e no projeto de sistemas embarcados.

2.2.1 Dificuldades Acadêmicas

Apesar de se mostrar um assunto extremamente interessante e com grandes possibilidades de estudo, os sistemas embarcados são pouco representados academicamente e em discussões públicas (Ryan, 1995, apud Marwedel, 2003, p.4). Na academia, um dos problemas de ensinar a projetar sistemas embarcados é o equipamento necessário para tornar o tópico interessante e prático. Além disso, sistemas embarcados reais são muito complexos e, portanto, difíceis de ensinar (Marwedel 2003).

Por outro lado, sistemas embarcados não aparecem como uma disciplina central no currículo de Ciência da Computação e nem no de Engenharia Elétrica. Na prática, engenheiros elétricos freqüentemente elaboram arquiteturas de softwares e cientistas da computação têm que lidar com restrições físicas. Ou seja, o projeto de sistemas embarcados está relacionado aos dois currículos. No entanto, existe uma barreira, iniciada na ainda academia, que separa estes dois mundos. E a indústria busca desesperadamente por engenheiros que possuam conhecimentos suficientes nos dois campos (Henzinger 2007).

Se em um extremo tem-se o fato de que as pesquisas em ciência da computação ignoraram por muito tempo os sistemas embarcados, pois utilizavam abstrações que de certa forma não levavam em conta as características físicas. No outro, tem-se a comprovação de que o projeto de sistemas embarcados vai além do conhecimento

tradicional dos engenheiros elétricos, uma vez que o cálculo e o software são partes integrantes dos sistemas embarcados (Henzinger 2007).

Segundo Lee (E. Lee, *Embedded Software* 2002), do ponto de vista do software, sistemas embarcados eram muito pequenos e retrógrados – com uso de suas técnicas peculiares como programação em linguagem *assembly*, e muito limitado pelos custos de hardware. Conseqüentemente, o projeto destes sistemas não foi beneficiado pela valiosa abstração do desenvolvimento do século XX. Tais sistemas não utilizam recursos como modelagem orientada a objetos, polimorfismo e gerenciamento automático de memória (E. Lee, *Embedded Software* 2002). As melhores tecnologias com seu uso desregrado de memória, camadas de abstração, algoritmos elaborados e otimizações estatísticas não pareciam aplicáveis. E como os resultados das pesquisas não resolviam os problemas dos sistemas embarcados, o problema não era, portanto, interessante aos olhos dos pesquisadores (E. Lee, *What's Ahead for Embedded Software?* 2000).

No entanto, estas deficiências têm se transformando em oportunidades. Os pesquisadores estão começando a reconstruir suas pesquisas de maneira que possam adequá-las aos problemas reais e diferenciados expostos pelos sistemas embarcados. E o problema mais urgente é, justamente, como adaptar as técnicas de software existentes para enfrentar os desafios do mundo físico (E. Lee, *What's Ahead for Embedded Software?* 2000).

2.2.2 Dificuldades no Projeto

2.2.2.1 Dificuldades Tecnológicas

O número crescente de aplicações resultou na necessidade de se construir tecnologias que suportem o projeto de sistemas embarcados. As tecnologias e ferramentas disponíveis atualmente ainda possuem limitações significativas, dentre elas a demanda por melhores linguagens de especificação, ferramentas de geração de código a partir das especificações, verificadores de tempo, sistemas operacionais de tempo real, tecnologias de baixo custo de energia e técnicas de projeto para sistemas confiáveis (Marwedel 2003).

Estas visões distintas entre as técnicas modernas de software e as necessidades dos sistemas embarcados não é uma novidade. Remontando-se às raízes da computação nota-se que as interfaces com o mundo real somente começaram a se estender recentemente, depois dos teclados e monitores. A computação tem suas origens na transformação de dados, não na interação com sensores, atuadores ou mesmo humanos. Entretanto, softwares em uma rede de sistemas embarcados serão formados, certamente, por componentes que operarão de maneira concorrente e em tempo real, com grande freqüência interagindo remotamente (E. Lee, *What's Ahead for Embedded Software?* 2000).

Ademais, o projeto de sistemas embarcados é extremamente complexo, por envolver conceitos até agora pouco analisados pela computação de propósitos gerais. Questões como portabilidade e limite de consumo de potência sem perda de desempenho, baixa disponibilidade de memória, necessidade de segurança e confiabilidade, possibilidade de funcionamento em uma rede maior, interação constante com o mundo real e o curto tempo de projeto tornam o desenvolvimento de sistemas computacionais embarcados uma área em si (Wolf 2001). Portanto, os desafios da pesquisa são enormes, uma vez que os prazos de entrega dos sistemas é cada vez menor, o custo e a complexidade desse tipo de sistemas é extremamente alta, não sendo confiável a utilização de metodologias de desenvolvimento com soluções *ad hoc*. Paradoxalmente, as tecnologias de engenharia de software não possuem características específicas para lidar com requisitos de memória, energia ou tempo real.

Uma das principais características dos sistemas embarcados, no entanto, é o fato de interagirem com o mundo real e, deste modo, não operarem em um ambiente estritamente controlado. Em consequência, este tipo de sistema apresenta características extremamente complexas, como:

- reação a eventos externos em tempo real;
- concorrência – sistemas embarcados raramente interagem com um único processo físico. Eles devem reagir, simultaneamente, a estímulos provenientes da rede e de uma variedade de sensores e, ao mesmo tempo, reter controle temporário sobre os atuadores;
- vivacidade (*liveness*) – esta é uma característica crítica, pois os programas não podem terminar ou ficar bloqueado enquanto esperam a ocorrência de eventos que nunca acontecerão;
- heterogeneidade – é uma característica intrínseca dos sistemas embarcados. Eles misturam estilos de computação e tecnologias de implementação. Primeiro porque tais sistemas são, freqüentemente, uma mistura de hardware e software, por isso, softwares embarcados interagem com hardwares projetados especificamente para interagir com eles. Sistemas embarcados também são heterogêneos no estilo de manipulação. Eles interagem com eventos ocorrendo irregularmente no tempo (alarmes, comandos do usuário, gatilhos de sensores, etc.) e regularmente no tempo (amostras de sensores de dados e sinais de controle de atuadores);
- reatividade – reagem continuamente com o ambiente na velocidade do ambiente. Sistemas reativos têm restrições de tempo-real e são freqüentemente de

segurança crítica, no sentido que falhas podem resultar na perda de vidas humanas. E eles nunca terminam e devem ser capazes de se adaptarem às mudanças. Tais sistemas estão sempre sendo redesenhados enquanto operam e não podem falhar;

- dentre outros já citados – adaptação a limites de tamanho e peso; gerenciamento de consumo de potência sem perda de desempenho; baixa disponibilidade de memória; satisfação a requisitos de confiabilidade e segurança e adequação a restrições de orçamento.

A partir do exposto, nota-se que outro grande desafio no desenvolvimento de sistemas embarcados é o atendimento de seus requisitos não-funcionais, ou seja, aqueles cuja preocupação não é com a funcionalidade do sistema. Independente do tipo de software, a especificação dos requisitos não-funcionais já é prejudicada pelo fato das técnicas adequadas para sua identificação e classificação não serem amplamente difundidas. Em sistemas embarcados este complicador é ainda maior, dada a imensa quantidade de requisitos não-funcionais a serem analisados.

2.2.2.2 Dificuldades do Negócio

Além das dificuldades técnicas, algumas dificuldades advindas do negócio também surgiram recentemente no projeto de sistemas embarcados, as quais os novos processos de desenvolvimento devem considerar (Sangiovanni-Vincentelli 2004): crescimento dos custos de engenharia não recorrente, elevando os custos da produção de um circuito integrado (CI); simultânea pressão para a redução do tempo de entrega (*time-to-market*) de um produto, e sua crescente complexidade; alteração de um modelo de negócio vertical para um modelo horizontal, onde a responsabilidade da produção de um novo projeto é distribuída entre companhias distintas.

Dado o exposto, nota-se que projetar um sistema embarcado é uma tarefa árdua. E o aumento das conexões, com a internet e entre os próprios dispositivos, introduz complicações significativas como a possibilidade de baixar módulos que reconfiguram dinamicamente o sistema. Além disso, os consumidores demandam funcionalidades cada dia mais elaboradas, que aumentam muito a complexidade do software. O fato é que tais sistemas não podem mais ser projetados por um único engenheiro dedicado a construir dezenas de *quilobytes* de código *assembly* (E. Lee, What's Ahead for Embedded Software? 2000).

Por outro lado, softwares embarcados freqüentemente encapsulam o conhecimento do domínio, especialmente quando é necessário processar dados de sensores ou controlar atuadores. Até mesmo programas muito pequenos podem conter algoritmos altamente

sofisticados, o que requer um profundo conhecimento do domínio e das tecnologias de suporte, como as de processamento de sinal.

2.2.2.3 Fracasso nos projetos

Um estudo realizado por (Carro 2002) com mais de 450 desenvolvedores de sistemas embarcados identificaram alguns fatores que contribuem para o aumento do número de projetos fracassados (atrasados ou cancelados):

- mudança na especificação;
- especificação incompleta;
- complexidade da aplicação;
- quantidade insuficiente de desenvolvedores, e
- tecnologias específicas para a aplicação de softwares embarcados.

Para resolver todas estas questões, as técnicas de projeto de software existentes não são apropriadas. Em parte porque são recentes, em outra, porque não levam em consideração as características específicas de sistemas embarcados. As tecnologias de desenvolvimento existentes não avaliam adequadamente o impacto em sistemas embarcados e nem são customizáveis a ponto de servirem a tais propósitos (Graaf 2003).

Por fim, em decorrência do conhecimento do domínio requerido: sistemas embarcados são projetados, freqüentemente, por engenheiros treinados classicamente no domínio em que trabalham. Os engenheiros de tais sistemas raramente são cientistas da computação, possuindo, portanto, pouco conhecimento de teoria da computação, concorrência, projeto orientado a objetos, sistemas operacionais e semântica (E. Lee, What's Ahead for Embedded Software? 2000). Na realidade, segundo (E. Lee, What's Ahead for Embedded Software? 2000) é discutível se outras disciplinas da engenharia teriam muito pouco a oferecer ao projeto de sistemas embarcados atualmente, devido a sua visão diferenciada a respeito da questão temporal e do uso desregrado de recursos de hardware. Mas estas disciplinas serão essenciais na medida em que os softwares embarcados forem se tornando cada vez mais complexos, modulares, adaptativos e conectáveis.

2.3 Considerações Finais

Apesar de serem muitas são as dificuldades enfrentadas no desenvolvimento de sistemas embarcados, este trabalho limitar-se-á a entender e avaliar os processos atuais de Engenharia de Requisitos deste tipo de sistema. O produto final deste estudo consistirá na

elaboração de um processo que se adapte de maneira adequada a este domínio de software.

Sendo assim, o capítulo seguinte abordará conceitos de processo de desenvolvimento de software, tanto para sistemas tradicionais quanto para sistemas embarcados, bem como técnicas para a modelagem de processos.

3. PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

O desenvolvimento de software de alta qualidade não é uma tarefa trivial desde o surgimento da indústria de software. No início os sistemas eram desenvolvidos sem a utilização de métodos sistemáticos e praticamente sem nenhuma documentação (*ad hoc*). O sucesso e a qualidade dos softwares dependiam diretamente dos profissionais envolvidos em seu desenvolvimento (Farias 2006).

Com o passar dos anos e com o aumento da complexidade dos sistemas, esse cenário começou a se modificar. Tornou-se evidente a necessidade de definir e utilizar métodos sistemáticos e ferramentas que guiassem o desenvolvimento de software, o que levou ao surgimento da engenharia de software (Pimentel 2007).

Os processos de desenvolvimento surgiram como parte da engenharia de software. Seu principal objetivo é assegurar a qualidade no desenvolvimento de sistemas em todas as suas etapas, para isso eles servem de guia para a execução de atividades, definem quais produtos serão desenvolvidos e quando, coordenam as mudanças necessárias, e auxiliam o acompanhamento do progresso do desenvolvimento do software (Pimentel 2007).

Este capítulo aborda conceitos relacionados aos processos de desenvolvimento de software. Sua organização é realizada da seguinte forma: na seção 3.1 será apresentada uma breve descrição de processos de desenvolvimento de software. A seção 3.2 evidencia alguns modelos de processos e os itens que os compõem. A seção 3.3 apresenta as fases e o fluxo de modelagem de um processo. Nas seções 3.4 e 3.5 são relatados processos de desenvolvimento de software tradicionais e específicos para o desenvolvimento de sistemas embarcados. Finalmente, a seção 3.6 descreve as considerações finais do capítulo.

3.1 Processo

Diversas etapas são executadas durante o processo de desenvolvimento de software. As atividades que as compõem iniciam com a identificação das necessidades do usuário, prosseguem com seu refinamento e finalizam com a disponibilização do software para sua utilização pelo usuário final. Esse conjunto de atividades que recebe dados de entrada realiza um processamento que adiciona valor a esses dados e produz uma saída de valor a um cliente específico é chamado de processo (Gonçalves 2000).

Pressman (Pressman 2006) afirma que um processo de software é caracterizado por um pequeno número de atividades que são aplicadas a qualquer tipo de projeto independente de seu tamanho e custo, e por um conjunto de tarefas. Este conjunto de tarefas é composto de coleções de atividades de engenharia de software; que podem ser adaptadas às necessidades da equipe e às características do projeto de software; marcos

de projeto, produtos de trabalho e pontos de garantia da qualidade. Suas atividades podem ser adaptadas às necessidades da equipe e às características do projeto de software.

Segundo Weber (Weber 2002), os processos de software têm o seu principal enfoque nos processos técnicos e gerenciais relacionados ao desenvolvimento de software cujo objetivo é: planejar, monitorar, gerenciar, executar, controlar e melhorar atividades relacionadas a software. Dessa forma, a correta definição do conjunto de processos das atividades da empresas, agrupados, documentados e institucionalizados subsidiam a geração de produtos com altos níveis de qualidade (J. C. Hauck 2004).

3.2 Modelo de Processo de Software

Conforme Feiler (Feiler e Humphrey 1993), o modelo de processo é uma representação abstrata da arquitetura, projeto ou definição do processo de software, que descreve, em diferentes níveis de detalhes, uma organização dos elementos de um processo e provê definições da maneira como devem ser realizadas a avaliação e a melhoria de processo.

Um modelo de processo é composto de grupos de processos e de sub-modelos que variam de acordo com as abordagens de diferentes autores. Acuña em (Acuña, et al. 2000) define que um processo é composto basicamente de dois sub-processos inter-relacionados e dois sub-modelos: o processo de produção, que consiste na construção e na manutenção do software; e o gerenciamento do processo, que é responsável por estimar, planejar e controlar os recursos necessário para executar o processo de produção, além dos sub-modelos de gerenciamento do processo e do próprio modelo de processo.

A ISO/IEC 12207 (*International Organization for Standardization / International Electrotechnical Commission*) é uma norma que define e classifica com maior nível de detalhe o grupo de processos que compõem um modelo de processo de software, abrangendo todo o ciclo de vida do software desde a concepção até sua descontinuidade (Calsalvara, et al. 2000). Os processos da ISO/IEC 12207 são agrupados de acordo com seu principal objetivo no ciclo de vida do projeto de forma que sua estrutura seja flexível, modular e adaptável conforme as necessidades de cada organização (Calsalvara, et al. 2000). A Figura 2 ilustra o modelo de processos da norma ISO/IEC 12207.

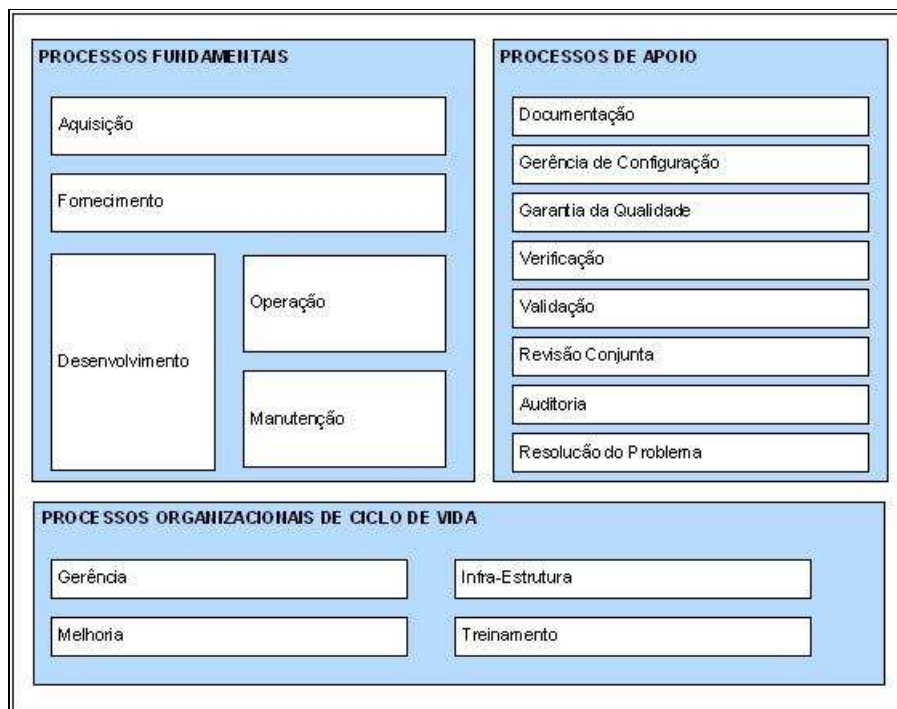


Figura 2 – Modelo de Processo ISO/IEC 12207 (Calsalvara, et al. 2000)

O modelo de processo da norma ISO/IEC 12207 é composto de três tipos de processos:

- **Processos Fundamentais:** São processos diretamente relacionados à produção do software em si.
- **Processos de Apoio:** servem como ferramenta de auxílio aos processos fundamentais. Os processos de apoio são de fundamental importância para o suporte a continuidade e a qualidade dos processos fundamentais.
- **Processos Organizacionais:** Viabilizam a manutenção dos demais processos da organização e sub-existem dentro das organizações independente de projetos específicos.

O objetivo do modelo de processo apresentado pela ISO/IEC 12207, e também por outros modelos de processo e de melhoria de processo, é que seja fornecida uma estrutura para que todos os envolvidos com o desenvolvimento do software utilizem uma linguagem comum na definição dos processos da organização (Calsalvara, et al. 2000).

3.2.1 Elementos do Modelo de Processo de Software




Os elementos que compõem o modelo de processo de software oferecem aos seus envolvidos um melhor entendimento de seu papel na organização, das funções desempenhadas no processo de desenvolvimento da empresa e de suas atividades ao





longo de todo o ciclo de desenvolvimento (Weber 2002). Visando representar de forma gráfica as características e os componentes de modelagem de um processo de desenvolvimento a OMG criou o SPEM *Metamodel* (OMG 2005).

O SPEM é um *profile* da notação UML que foi proposto pela OMG (*Object Management Group*) para a descrição de processos de desenvolvimento de software e de seus componentes. A proposta inicial da OMG limita-se a definir o conjunto mínimo de processo que modela os elementos necessários para descrever qualquer processo de desenvolvimento de software, sem adicionar modelos específicos ou condições para qualquer área específica ou disciplina, como gerenciamento de projeto ou análise.

Segundo o SPEM, um modelo de processos inclui os elementos de processo definido na Tabela 1 (OMG 2005):

Tabela 1 - Componentes do SPEM

Estereótipo/Conceito	Descrição (Adicionar a descrição das figurinhas)	Notação (colocar as figurinhas)
Fase	É tempo entre os maiores <i>milestones</i> do projeto. Entre a execução das fases objetivos e metas são definidos, artefatos são criados ou complementados e decisões são tomadas para a mudança de fase.	
Disciplina	É uma coleção de atividades na qual está relacionada uma maior área de conhecimento.	
Atividade	Etapa de um processo que produz alterações de estado externamente visíveis no produto de software. Uma atividade pode ser representada por uma entrada, uma saída e um ou mais artefatos.	
Papel	É uma função dentro do processo, ou seja, é um grupo de responsabilidades,	

	privilégios e habilidades que são requeridas para a execução de terminada atividade;	
Produto de Trabalho	É um artefato consumido, produzido ou alterado durante a execução de uma atividade. São os subprodutos do processo, as entradas e saídas de uma atividade durante o processo;	
Orientações	São documentos que provêm um maior detalhe de informações sobre os diversos detalhes do processo.	
Documento	É um tipo de produto de trabalho que representa um arquivo texto.	
Modelo UML	É um tipo de produto de trabalho que representa um arquivo texto.	

A Figura 3 demonstra como os elementos de um modelo se inter-relacionam para formar um processo de desenvolvimento de software (J. C. Hauck 2004):

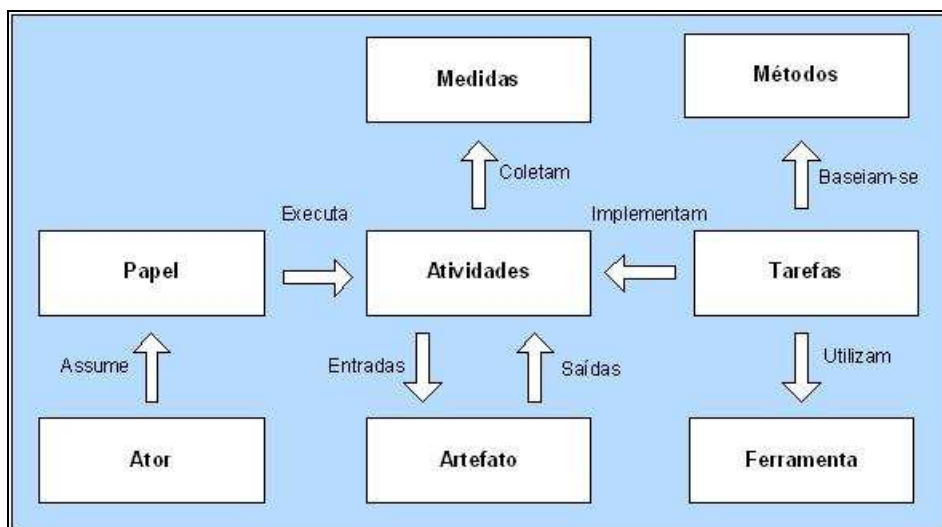


Figura 3 - Inter-relação entre os elementos de um modelo de processo de software (Weber 2002)

3.3 Modelagem de Processo

A modelagem de um processo é uma estrutura que descreve como identificar, documentar e institucionalizar as atividades envolvidas na construção de produtos de software. As pesquisas na área de modelagem de processos propõem diversos paradigmas para sua execução, no entanto, duas formas de modelagem de processos são amplamente utilizadas atualmente: a modelagem descritiva, que basicamente descreve o processo que está sendo analisado e a modelagem prescritiva, que propõem um modelo de processos ideal para a organização que está sendo analisada (J. C. Hauck 2002) (Mcchesney 1995).

Para que seja possível documentar, institucionalizar ou até mesmo propor melhorias em um processo é necessário que haja o entendimento das atividades, papéis envolvidos e de todos os artefatos produzidos na execução do processo. A modelagem descritiva é uma ferramenta utilizada para identificar como as atividades de desenvolvimento são executadas e quais as características envolvidas nos processos de desenvolvimento da organização levando em conta o contexto na qual ela está inserida (Weber 2002). A obtenção do modelo descritivo detalha formalmente como as atividades são realizadas no momento em que a avaliação do processo está sendo executada sem interferência direta na forma como elas são ou devem ser feitas. Possibilitando uma visão real do processo atual que permite uma futura análise para identificação de erros ou proposição de melhorias (Mcchesney 1995).

Após a avaliação dos processos das organizações em muitos casos são constatadas identificações de erros e proposição de melhorias, pois o processo modelado não atende a recomendação de normas de padronização e melhoria de processos. Nesse caso é necessário reformular ou readequar suas atividade e papéis. A modelagem prescritiva contempla os modelos prescritos após a avaliação dos processos institucionalizados nas organizações, pois descrevem como o software deveria ser desenvolvido (J. C. Hauck 2002). A modelagem de processos prescritiva pode ser executada de forma manual – baseadas em normas, padrões, melhores práticas ou metodologias, ou de forma automática – apoiada por ferramentas com modelos de processos automáticos (Mcchesney 1995).

Diversas são as classificações de modelagem de processos de software que são propostas na literatura, no entanto, as etapas da modelagem de processos são customizadas de acordo com a realidade vivenciada pelas empresas desenvolvedoras de software.

No caso das MPEs, que são o foco deste trabalho, na maioria dos casos os processos de desenvolvimento não são documentados e nem conhecidos pelos participantes da equipe. Dessa forma, é necessário que seja realizada uma avaliação inicial da cultura e do processo existente na organização seguida da etapa de elicitação (J. C. Hauck 2002). Após essas fases modelos prescritos são sugeridos tomando como base os

processos avaliados e a literatura que trata a padronização de processos. Após a prescrição do modelo, a documentação e a disseminação do processo devem ser realizadas (J. C. Hauck 2004). Essa mesclagem das características da modelagem descritiva e prescritiva gera um modelo de processo híbrido que utiliza as melhores características dos dois modelos com o objetivo de definir um processo de desenvolvimento otimizado que atinja altos níveis de qualidade levando em conta o contexto no qual a organização está inserida (J. C. Hauck 2002).

3.4 Modelos de Processos de Desenvolvimento de Software

3.4.1 Processo Unificado

O processo unificado (PU) é o produto final de três décadas de trabalho. Seu desenvolvimento partiu do *Objectory Process* (1987), via *Rational Objectory Process* (1997) até o *Rational Unified Process* (RUP)(1998). Neste percurso o processo sofreu as mais diversas influências, sendo as abordagens de maior impacto as adotadas pela Ericsson e Rational.

O PU surgiu com o objetivo de definir um processo de desenvolvimento de software visando à construção de sistemas orientados e a objetos (Larman 2004). Ele combina as melhores práticas estudadas por Booch, Jacobson e Rumbaugh e tem como base a *Unified Modeling Language* (UML). Entretanto, as características que o torna realmente único podem ser resumidas em três aspectos: é dirigido a caso de uso, centrado na arquitetura e iterativo e incremental (Jacobson, Boock e Rumbaugh 1999).

No PU o caso de uso (UC) é o ponto central do desenvolvimento. Ele captura os requisitos funcionais do sistema e força os engenheiros de software a pensarem em termo do valor para o usuário e não apenas nas funcionalidades que o sistema deve possuir. Além disso, os UCs dirigem todo o projeto de desenvolvimento de software desde seu projeto, até sua implementação e testes (Jacobson, Boock e Rumbaugh 1999).

A arquitetura de software no PU descreve as diferentes visões do sistema que está sendo construído. Ela incorpora os aspectos estáticos e dinâmicos mais significantes para o sistema e reflete além dos casos de uso, as necessidades da empresa e do cliente. A arquitetura também é influenciada por fatores como a plataforma de software em que o sistema executará, os blocos reusáveis disponíveis, considerações de implantação, sistemas legados e requisitos não-funcionais (Jacobson, Boock e Rumbaugh 1999).

Os casos de uso e a arquitetura devem ser desenvolvidos em paralelo e são interligados na medida em que o produto deva possuir função e estrutura. A função corresponde ao UC e a estrutura à arquitetura. Se por um lado os casos de uso, quando

realizados, devem estar de acordo com arquitetura definida. Por outro, a arquitetura deve permitir a realização de todos os casos de uso (Jacobson, Boock e Rumbaugh 1999).

Além de usar a UML e ser baseado em casos de uso, arquitetura e desenvolvimento iterativo e incremental, o PU é baseado em componentes. E para que esta abordagem funcione é necessário um processo que leve em consideração ciclos, fases, fluxos de trabalho, mitigação de riscos, controle de qualidade, gerenciamento de projetos e controle de configuração. O PU estabelece uma *framework* que integra todas estas diferentes características em um único modelo de processos customizável (Jacobson, Boock e Rumbaugh 1999).

3.4.2 Rational Unified Process

Construído com base no PU o RUP é um processo de engenharia de software lançado no mercado em 1996 para dar suporte às atividades relacionadas ao ciclo de vida do processo de desenvolvimento (Barros, Esmeraldo e Silva, Uma Metodologia de Desenvolvimento Baseada em Componentes Aplicada à Modelagem de Sistemas Embarcados 2006) (Kruchten 2003). Ele descreve todos os artefatos, atividades e papéis, fornece diretrizes e inclui modelos para a maioria dos artefatos (Larman 2004).

O principal objetivo do RUP é garantir a produção de softwares de alta qualidade que atenda às necessidades dos usuários dentro de um orçamento e de um cronograma previsto (Kruchten 2003).

Assim como estabelecido no PU, três princípios básicos fundamentam o RUP: orientação a casos de uso, arquitetura e iteração. O RUP utiliza uma abordagem baseada em fases e disciplinas e atribui tarefas e responsabilidades dentro de uma organização desenvolvedora de software (Kruchten 2003). A Figura 4 ilustra as duas dimensões abordadas pelo RUP:

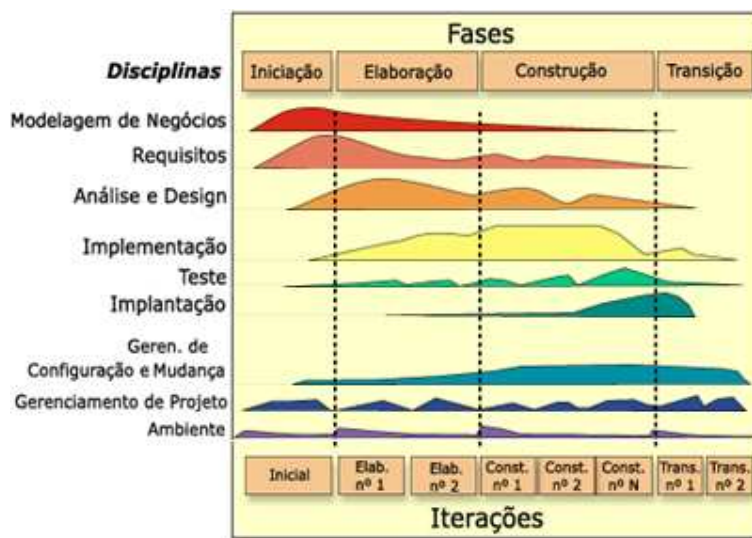


Figura 4 - Gráfico das Baleias do RUP (Kruchten 2003)

O eixo horizontal representa o tempo e mostra os aspectos do ciclo de vida do processo à medida que se desenvolve. Representa os aspectos dinâmicos do processo e é descrito por meio de fases, iterações e marcos. O eixo vertical representa as disciplinas, que agrupam as atividades de maneira lógica, por natureza, representa os aspectos estáticos do processo e é descrito por meio de componentes, disciplinas, atividades, fluxos de trabalho, artefatos e papéis. Segundo Kruchten (Kruchten 2003), o RUP pode ser utilizado em diversos tipos de projetos e em empresas de pequeno, médio e grande porte, devido a sua flexibilidade e facilidade de configuração.

3.5 Modelos de Processo de Desenvolvimento de Software Embarcado

Conforme descrito no Capítulo 02 (Sistemas Embarcados), os problemas do desenvolvimento de software tradicional são diferentes dos problemas de desenvolvimento de sistemas embarcados no qual um conjunto mais amplo de requisitos compõe o produto final (Barros, et al. 2005). O ponto crucial do desenvolvimento de sistemas embarcados é a determinação dos muitos desafios da integração entre software e hardware. Para isso, diversos modelos de processos para sistemas embarcados foram propostos na literatura. As seções seguintes apresentam alguns desses modelos.

3.5.1 RUPSE

Lançado no mercado em 2001, o *Rational Unified Process for Systems Engineering* (RUPSE) é uma ferramenta que oferece às organizações desenvolvedoras de software uma estrutura de suporte para facilitar a definição do ciclo de vida de engenharia de sistemas (M. Cantor 2003) (Cantor, Rational Unified Process for Sytem Engerrring - Análise e Design de

Requisitos 2003). Diferente do RUP, que foca no suporte ao desenvolvimento de software, o RUPSE foca no suporte à Engenharia de Sistemas (M. Cantor 2003).

O RUPSE oferece suporte ao desenvolvimento de sistemas em grande escala, compostos de software, hardware, trabalhadores e componentes de informações (M. Cantor 2003). Ele fornece soluções que inclui um modelo de arquitetura que permite a abordagem de um conjunto diferente de perspectivas (lógicas, físicas, informação etc.) (M. Cantor 2003)

Segundo Cantor (M. Cantor 2003), o RUPSE determina projetos que:

1. Necessitem de diversas equipes com desenvolvimento simultâneo devido a seu tamanho.
2. Desenvolvam simultaneamente hardware e software.
3. Sejam complexos o suficiente para possuírem problemas de implementação arquiteturalmente significativos.

Tanto no RUP como no RUPSE, diversos papéis são envolvidos no processo de desenvolvimento, esses papéis são desenvolvidos por pessoas que compõem a equipe de desenvolvimento como analistas de requisitos, arquitetos, projetistas, desenvolvedores entre outros. Além disso, o ciclo de vida, as iterações e as disciplinas do RUPSE são herdadas do RUP. No entanto, para suportar a especificação de novas visões da engenharia de sistemas, o RUPSE inclui novos artefatos, atividades e funções, altera a estrutura de desenvolvimento do RUP (estrutura 4+1) e utiliza não só a UML 2.0, mas também seus estereótipos (Cantor, Rational Unified Process for System Engineering - Análise e Design de Requisitos 2003).

A especificação do sistema no RUPSE estuda como se espera que o sistema execute em um determinado contexto. Ou seja, são analisadas características como as funcionalidades do sistema que são externamente visíveis, os serviços fornecidos e as medidas de eficácia que devem ser atendidas. Dessa forma, é realizada uma análise da visão e da missão da empresa e da função do sistema nesse contexto (M. Cantor 2003).

A análise do ambiente no qual o sistema está envolvido é modelada em diagramas de contexto que estabelecem seu escopo, limites, serviços, atributos e comunicações com outras entidades. Esses diagramas utilizam estereótipos da linguagem UML para sua representação gráfica (M. Cantor 2003).

3.5.2 IPProcess

O Processo de Desenvolvimento de Software (IPProcess) surgiu da necessidade de criação de uma metodologia que resolvesse os problemas específicos do desenvolvimento

de software embarcado (Barros, Esmeraldo e Silva, Uma Metodologia de Desenvolvimento Baseada em Componentes Aplicada à Modelagem de Sistemas Embarcados 2006). Para isso, a abordagem de desenvolvimento baseados em componente (DBC) e a utilização de disciplinas na atribuição de tarefas e responsabilidade dentro de uma organização foram unificadas em um modelo de desenvolvimento de sistemas embarcados embarcado (Barros, et al. 2005).

Dessa forma, o IPProcess propõe a utilização de técnicas da Engenharia de Software na modelagem de sistemas de software iniciando em uma descrição de hardware em alto nível que permite a detecção de erros na fase de Projeto enfocando a qualidade no final da implementação (Barros, et al. 2005).

A Figura 5 ilustra a arquitetura do IPProcess sob a perspectiva de fases e disciplinas (Barros, et al. 2005):

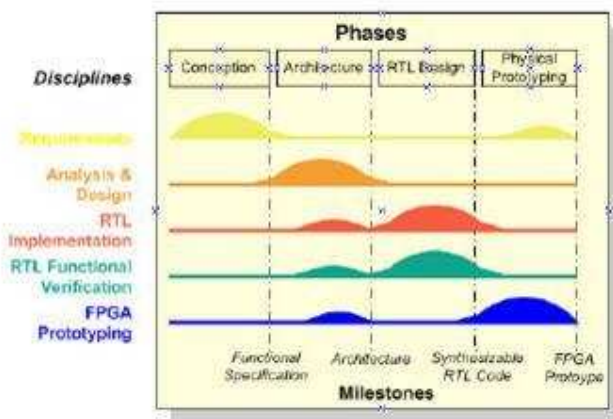


Figura 5 - Gráfico das Baleias do IPProcess (Barros, Bione, et al. 2005)

As fases do processo são representadas no eixo horizontal e são decompostas seqüencialmente ao longo do tempo: especificação funcional, arquitetura, projeto RTL e prototipação física (Barros, et al. 2005). As duas fases iniciais descrevem o entendimento do problema (o que deve ser feito) e sua modelagem (como deve ser feito) utilizando a notação UML e a UML Real Time. O processo de implementação é efetivamente iniciado na fase de Projeto RTL, pelo fato de que é necessário um período de tempo para a discussão exaustiva das funcionalidades, dos requisitos e das possíveis arquiteturas que serão utilizadas. A seguir será descrito um breve resumo de cada uma das fases do IPProcess (Barros, Esmeraldo e Silva, Uma Metodologia de Desenvolvimento Baseada em Componentes Aplicada à Modelagem de Sistemas Embarcados 2006):

4. **Concepção:** Tem como objetivo definir o escopo do sistema (ou componente) encontrando, entendendo e definindo seus requisitos funcionais e não-funcionais.

5. Arquitetura: Nesta fase são definidos os componentes da arquitetura e suas interfaces e é construída uma prova arquitetural que demonstra que a arquitetura está estável e suporta os requisitos identificados na fase de concepção.
6. Projeto : O propósito dessa fase é o desenvolvimento de um modelo de simulação baseado na arquitetura do sistema que é utilizado para realizar a verificação dos componentes da arquitetura.
7. Prototipação Física: Nesta fase são construídos os protótipos físicos que garantam que o sistema pode ser disponibilizado para seus usuários finais.

O IPProcess é composto pelas seguintes disciplinas (Barros, et al. 2005):

8. Requisitos: A disciplina de requisitos possui um maior esforço de execução na fase de concepção, com isso, as principais atividades da disciplina são a análise do problema, o entendimento das necessidades dos usuários e a definição de interfaces externas levando em conta os objetivos e as necessidades dos usuários. Os artefatos produzidos na disciplina de requisitos são o Documento de Visão, o Glossário, o Documento de Requisitos do Usuário, o Modelo de Caso de Uso e o Documento de Especificação de Requisitos
9. Análise e Projeto: O principal objetivo da disciplina de análise e projeto é a elaboração do projeto da arquitetura do sistema e associação de funcionalidades aos seus componentes, portanto, grande parte do esforço dessa disciplina se concentra na fase de arquitetura do sistema.
10. Verificação Funcional: Esta disciplina do IPProcess visa validar se os requisitos foram implementados de forma correta e encontrar defeitos no modelo de simulação, para isso, um plano de verificação é elaborado juntamente com a execução dos testes.
11. Prototipação FPGA: O principal propósito desta disciplina é transformar o modelo de simulação em um protótipo físico e validar se os requisitos foram corretamente implementados nesse protótipo.

3.5.3 Processo Unificado para Sistemas Embarcados (UPES)

Outro modelo de processos que leva em consideração as características específicas dos sistemas embarcados é o Processo Unificado para Sistemas Embarcados (UPES). O UPES norteia o desenvolvimento de sistemas embarcados desde a concepção dos requisitos até a efetiva implementação do sistema (Riccobene, et al. 2007).

O UPES baseia-se na abordagem MDA (*Model Driven Architecture*), onde a aplicação é gerada automaticamente por meio de modelos UML (*Unified Model Language*). Seu ciclo de desenvolvimento em Y ilustrado na Figura 6 (Riccobene, et al. 2007). Em um lado o processo PU tradicional é utilizado na definição de um modelo executável da aplicação, em outro lado é disponibilizada a plataforma de hardware modelada em uma linguagem própria para sistema embarcados (UML 2.0 e suas extensões) (Riccobene, et al. 2007). A intersecção entre os dois lados une o mapeamento do modelo da aplicação considerando o modelo da plataforma (Riccobene, et al. 2007). Essa junção é realizada por meio da elaboração de um modelo de mapeamento que refina os requisitos de entrada em termos de diagramas de componentes e de construção (Riccobene, et al. 2007).

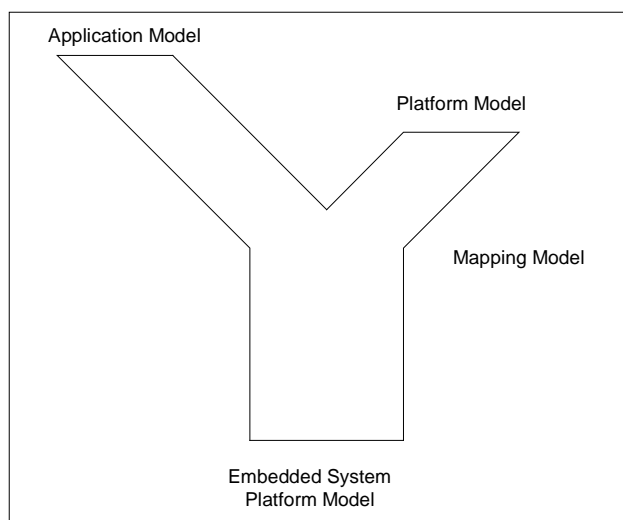


Figura 6 - Gráfico Y do Modelo UPES

Os componentes de hardware são mapeados diretamente no modelo de plataforma e os componentes de software implementados como tarefas que usam serviços providos pelo modelo de aplicação (Riccobene, et al. 2007).

O Processo de Engenharia de Requisitos deste modelo foca especificamente no Processo Unificado Tradicional, dessa forma, não foi incluído nesse modelo nenhuma adaptação para o contexto de sistemas embarcados.

3.6 Considerações Finais do Capítulo

Este capítulo aborda, conceitos relacionados a processos de desenvolvimento de software e seus modelos apresentados na literatura, além de técnicas de modelagem de processos, suas fases e componentes. Para finalizar, são descritos alguns processos de desenvolvimento de software tradicionais e específicos para sistemas embarcados.

A Tabela 2 apresenta uma análise comparativa dos processos de desenvolvimento de sistemas embarcados que visa avaliar a efetiva abrangência das características desejadas em um processo de engenharia de requisitos (ER).

Tabela 2 - Comparativo das Características de ER em Processos de Desenvolvimento de Sistemas Embarcados

Característica do Processo de ER	RUP SE	IPProcess	UPES
1. Há um processo de Engenharia de Requisitos definido?	S ¹	S	S
2. A atividade de levantamento de requisitos é realizada?	S	S	S
3. A atividade de especificação se preocupa somente com o que será desenvolvido?	S	S	S
4. A atividade de design é tratada separada da especificação?	N ²	S	S
5. O processo de engenharia de requisitos possui características específicas do desenvolvimento de sistemas embarcados	S	N	N
6. O processo de engenharia de requisitos possui a mesma estrutura dos processos tradicionais?	N	S	S

Conforme foi observado na descrição do capítulo e no comparativo da Tabela 2, os processos de desenvolvimento de software embarcado se assemelham aos processos tradicionais. Em muitos deles, o maior enfoque é dado no projeto software ficando a engenharia de requisitos sem adequações necessárias ao desenvolvimento desse tipo de sistemas. Dessa forma, é identificado como oportunidade de pesquisa, o estudo das adaptações necessárias aos processos existentes para que o processo de engenharia de requisitos possa atender as características específicas do software embarcado.

O capítulo seguinte irá apresentar um detalhamento dos processos de requisitos dos processos de desenvolvimento propostos na literatura. Serão abordados conceitos relacionados aos processos de requisitos tradicionais e específicos para o desenvolvimento de sistemas embarcados

4. REQUISITOS

Um dos maiores desafios no processo de desenvolvimento de software ainda é atender as expectativas dos usuários. Entender os requisitos de um problema está entre as tarefas mais difíceis enfrentadas por um Engenheiro de Software (Pressman 2006). Por mais bem definido e otimizado que seja o sistema é necessário esforço, atenção e experiência por parte da equipe de projeto para entregar ao cliente o que ele deseja e, realmente, precisa.

O ponto de partida na concepção de sistemas, sejam eles embarcados ou não, é a definição de seus requisitos funcionais e não-funcionais. De acordo com (Hofmann 2001), requisitos deficientes ainda são a maior causa de falhas nos projetos de software. Capers Jones descobriu, por meio de pesquisas realizadas com centenas de organizações, que a engenharia de requisitos é deficiente em mais de 75% das empresas (Jones, 1996, apud Hofmann, 2001, p. 58). Um estudo realizado por (Gastaldo 2003) demonstrou que cerca de 50% do retrabalho referente ao processo de desenvolvimento de software ocorre nas fases iniciais de elicitação, análise e documentação de requisitos, conforme Figura 7.

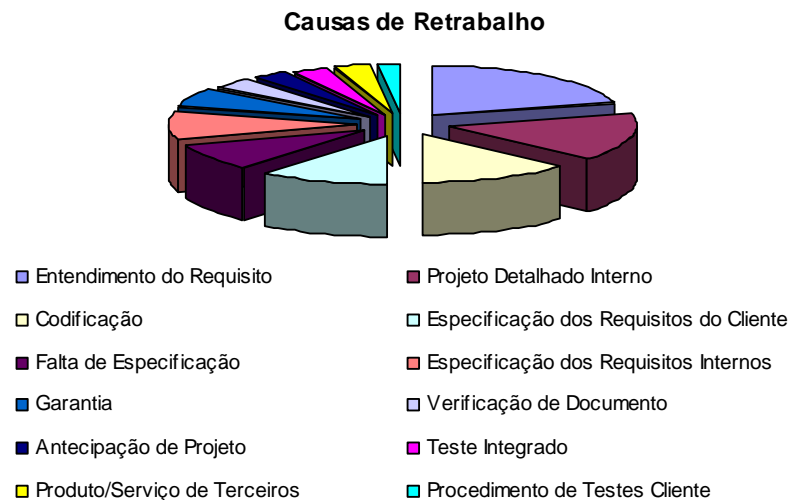


Figura 7 - Causas de Retrabalho (Gastaldo 2003)

Segundo Gastaldo (Gastaldo 2003), grande parte das causas do retrabalho é relacionada aos requisitos não-funcionais de desempenho. Sua pesquisa revelou que por não serem levados em consideração desde o início do projeto, estes requisitos acabam não

sendo atendidos. Em consequência de mudanças de estratégia a fim de inserir um requisito não-funcional não projetado, há aumento de custo e prazo de entrega.

Em sistemas embarcados a situação não tem se mostrado diferente. Conforme citado no capítulo 2 mudanças na especificação do software e especificações incompletas estão entre as maiores causas de fracasso nos projetos. Diante deste contexto, a Engenharia de Requisitos tem se tornado cada vez mais necessária para resolver os problemas encontrados nas organizações com relação à definição de sistemas. Elaborar um processo que preencha as lacunas existentes nas fases de elicitação, análise, documentação e validação de requisitos, especialmente os não-funcionais, é uma tarefa de grande importância para o sucesso dos projetos de sistemas embarcados.

Na seção 4.1 serão apresentados os conceitos básicos relacionados a requisitos de software. A seção 4.1.3 descreverá os aspectos de requisitos para sistemas embarcados. Na seção 4.3 podem ser encontradas algumas características importantes no processo de Engenharia de Requisitos para sistemas embarcados. Já a seção 4.4 abordará as considerações finais do capítulo.

4.1 Conceitos Básicos

Nas seções seguintes são apresentados alguns conceitos julgados importantes no contexto este estudo. Vale destacar que não foram descritos todos os conceitos relacionados à Engenharia de Requisitos, mas apenas aqueles considerados fundamentais para o bom entendimento da pesquisa.

4.1.1 Requisitos

Um requisito é definido como uma propriedade que o sistema deve apresentar a fim de resolver algum problema do mundo real (IEEE 2001). Já Sommerville (Sommerville 1998) entende requisitos como sendo descrições de como os sistemas devem se comportar, informações sobre o domínio de aplicações, restrições sobre o funcionamento do sistema, ou especificações de propriedades ou atributos do sistema. Ainda segundo ele, os requisitos são definidos durante os primeiros estágios do desenvolvimento de sistemas como uma especificação do que deve ser implementado.

Conforme (Sommerville 1998) um requisito pode descrever:

1. Uma funcionalidade em nível de usuário (ex.: o processador de texto Word deve incluir um revisor de escrita);

2. Uma propriedade bem geral (ex.: o sistema deve garantir que nenhuma informação pessoal estará disponível sem autorização);
3. Uma restrição específica no sistema (ex.: o sistema deve checar a chegada de mensagens 10 vezes por hora);
4. Como realizar algum cálculo (ex.: o cálculo do salário deve ser feito de acordo com a fórmula = (remuneração + comissão) – descontos);
5. Uma restrição no desenvolvimento do sistema (ex.: o sistema deve ser desenvolvido em C++).

Sendo assim, requisitos invariavelmente contêm uma mistura de informações de problemas, declarações de comportamento e propriedades do sistema e restrições de projeto e construção. Isto pode causar dificuldades porque as restrições de projeto e construção podem conflitar com outros requisitos. Contudo, esta é uma realidade da Engenharia de Requisitos, então, o Processo de Engenharia de requisitos deve incluir atividades para encontrar e resolver os problemas encontrados (Sommerville 1998).

4.1.2 Requisitos Funcionais x Requisitos Não-Funcionais

Segundo (IEEE 2001), os requisitos podem variar em seu objetivo e no tipo de propriedades que representam, podendo, desta forma, ser classificados em:

- Requisitos Funcionais: são aqueles que expressam funções ou serviços que um software deve ou pode ser capaz de executar ou fornecer. As funções ou serviços são, em geral, processos que utilizam entradas para produzir saídas (Cysneiros 2001).
- Requisitos Não-Funcionais: são requisitos que colocam restrições sobre o produto em desenvolvimento, sobre o processo de desenvolvimento, e também, especificam restrições externas ao produto (Figura 8). Normalmente são determinados pela natureza e tamanho do sistema no qual o software está inserido.

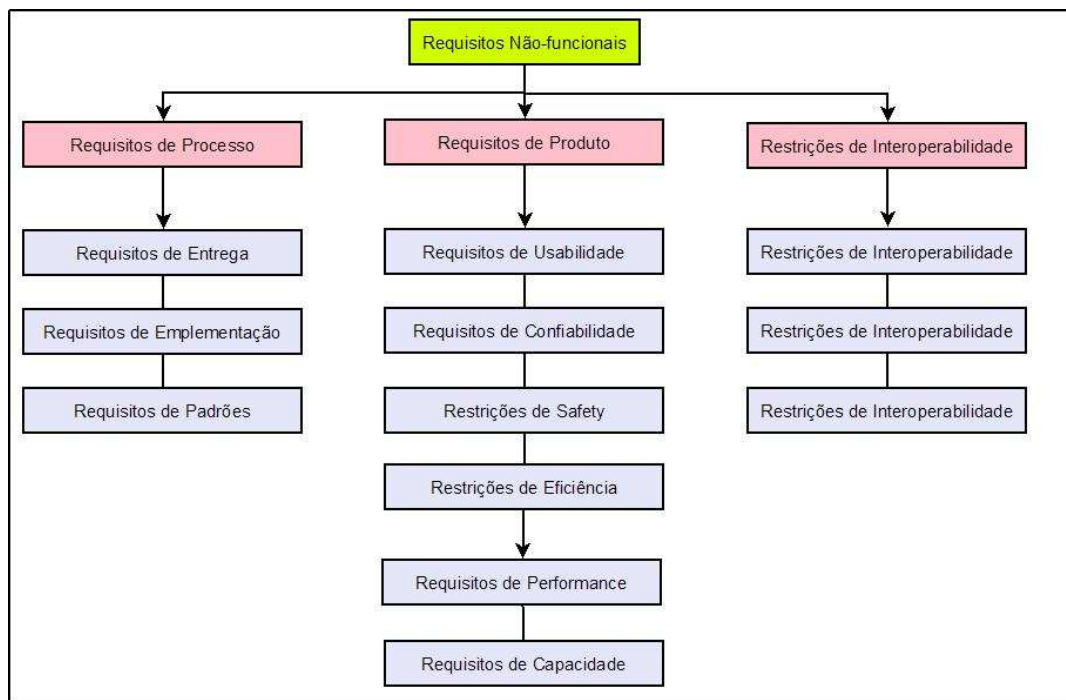


Figura 8 – Classificação de Requisitos Não-Funcionais (Sommerville 1998)

A distinção entre estes dois tipos de requisitos nem sempre é muito clara. Parte da razão advém para o fato de que os requisitos não-funcionais estão sempre relacionados a um requisito funcional (Chung 1999). De uma maneira geral, pode-se dizer que um requisito funcional expressa algum tipo de transformação que tem lugar no software, enquanto um requisito não-funcional expressa como essa transformação irá se comportar ou que qualidades específicas ela deverá possuir (Cysneiros 2001).

4.1.3 Engenharia de Requisitos

A Engenharia de Requisitos (ER), uma subárea da Engenharia de Software, estuda o processo de definição dos requisitos que o software deverá atender. O processo de definição de requisitos é uma interface entre os desejos e necessidades dos clientes e a posterior implementação desses requisitos em forma de software (Silva, Implantação de Melhoria no Processo "Engenharia de Requisitos" na Empresa Fórmula Informática 2004).

Segundo (Sommerville 1998), a ER é um termo relativamente novo que foi inventado para cobrir todas as atividades envolvidas no descobrimento, documentação e construção de uma série de requisitos para um sistema baseado em computação. O uso do termo "Engenharia" implica que técnicas sistemáticas e repetidas devem ser usadas para assegurar que os requisitos do sistema são completos, consistentes, relevantes etc.

A ER é a primeira e mais importante atividade de cunho técnico no desenvolvimento de um software (Kondo, Estudo de Requisitos do Software Embarcado no Segmento da Telemedicina 2006). No entanto, ela trata de conhecimentos não apenas técnicos, mas também gerenciais, organizacionais, econômicos e sociais (Silva, Implantação de Melhoria no Processo "Engenharia de Requisitos" na Empresa Fórmula Informática 2004). A ER ajuda os engenheiros de software a compreenderem melhor o problema que trabalharão para resolver, ou seja, o que o cliente quer, como os usuários finais vão interagir com o software e qual o impacto do software sobre o negócio (Pressman 2006).

As atividades da ER são realizadas não apenas pelos engenheiros de software, conhecidos também como engenheiros de sistema ou analistas, mas também por outros interessados (gerentes, clientes e usuários finais). Todos podem e devem participar da ER para garantir que o sistema a ser construído refletirá o desejo dos interessados. Afinal de contas, projetar e construir um software elegante que resolva o problema errado não serve às necessidades de ninguém. Essa é a razão pela qual é importante entender o que o cliente deseja antes de começar a projetar e construir um sistema baseado em computador (Pressman 2006).

Pesquisas realizadas até o momento sugerem que para grandes sistemas de hardware/software, aproximadamente 15% do orçamento total é gasto em atividades de ER (Sommerville 1998). Para sistemas pequenos, que normalmente são apenas software, o custo com requisitos normalmente é menor, cerca de 10% do total do orçamento (Sommerville 1998). O custo com requisitos não é baixo, no entanto, as conseqüências de requisitos errados são ainda maiores:

1. O sistema pode ser entregue fora do prazo e com custo superior ao estimado inicialmente;
2. O cliente e os usuários finais podem não ficar satisfeitos com o sistema. Podendo não utilizar algumas funcionalidades ou mesmo decidirem deixar o sistema de lado por completo;
3. O sistema pode ser de uso não confiável, com erros regulares e com desastres interrompendo a operação normal;
4. Se o sistema continuar em uso, o custo de evolução e manutenção normalmente são altos.

Vale destacar, ainda, que o custo para consertar erros de requisitos é, normalmente, muito menor do que consertar erros em estágios posteriores do processo de desenvolvimento. Corrigir problemas de requisitos podem demandar um retrabalho no projeto, implementação e teste do sistema. Conseqüentemente, os custos serão altos. Estima-se que o custo para corrigir erros de requisitos pode ser 100 vezes maior do que corrigir um simples erro de programa (Sommerville 1998).

1.1.1.1 Melhores Práticas na ER

De acordo com os estudos de Hofmann (Hofmann 2001), projetos de sucesso têm uma correta combinação de conhecimento, recursos e processos. A Tabela 3 resume a relação entre estes três elementos. Esta tabela apresenta as sugestões de melhores práticas, de acordo com cada uma das áreas foco (conhecimento, recursos ou processo), o custo de execução desta prática (em termos de introdução da prática e de sua aplicação) e os benefícios esperados da realização de tais práticas.

Tabela 3 - Melhores Práticas da ER (Hofmann 2001)

Área Foco	Melhores Práticas	Custo de Introdução	Custo de Aplicação	Benefícios Chaves
Conhecimento	Envolver clientes e usuários durante a ER	Baixo	Moderado	Melhor entendimento das reais necessidades
	Identificar e consultar todas as origens de requisitos	Baixo a Moderado	Moderado	Aprimoramento da cobertura dos requisitos
	Atribuir as atividades de ER a gerentes de projetos e membros da equipe habilidosos	Moderado a Alto	Moderado	Performance mais previsível
Recursos	Alocar de 15 a 30% do total de esforço do projeto para as atividades de ER	Baixo	Moderado a Alto	Manter uma alta qualidade de especificação do início ao fim do projeto
	Prover modelos e exemplos de especificação	Baixo a Moderado	Baixo	Aperfeiçoar a qualidade da especificação
	Manter um bom relacionamento com os envolvidos	Baixo	Baixo a Moderado	Satisfazer melhor as necessidades dos clientes
Processos	Priorizar requisitos	Baixo	Baixo a Moderado	Foco da atenção nas necessidades mais importantes dos clientes

	Desenvolver modelos complementares juntamente com os protótipos	Baixo a Moderado	Moderado	Eliminar inconsistências e ambigüidades na especificação
	Manter a matriz de rastreabilidade	Moderado	Moderado	Explicitar a ligação entre requisitos e produtos de trabalho
	Usar revisão por pares, cenários e "walkthroughs" para validar e verificar os requisitos	Baixo	Moderado	Especificação mais acurada e alta satisfação do cliente

Segundo o estudo realizado por Hofmann (Hofmann 2001), a interação com os envolvidos tem um papel decisivo do início ao fim dos projetos de sucesso em ER. Os times de mais sucesso sempre abarcam seus clientes e usuários no processo de ER e, além disso, mantêm um excelente relacionamento com todos os envolvidos. Ainda de acordo com as pesquisas de Hofmann (Hofmann 2001), a participação dos usuários é o fator mais importante para o sucesso da ER.

Além disso, boas equipes de ER também identificam as fronteiras do domínio da aplicação e dos principais envolvidos. E para validar seu entendimento do domínio da aplicação eles identificam e consultam todo tipo de fonte de requisitos: examinam artefatos de sistemas, materiais do sistema atual e dos anteriores etc.

Outro fator de grande importância, segundo (Hofmann 2001), é a escolha da equipe de requisitos. É necessário escolher membros com habilidades no domínio da aplicação, na tecnologia a ser utilizada e no processo de ER. Também é indispensável a escolha de gerentes de projetos capacitados para a ER. E, quando julgarem necessário, consultar especialistas do domínio e envolvidos, a fim de aumentarem o conhecimento da equipe.

Projetos considerados de sucesso também alocam uma significativa parcela de recursos para a ER. Conforme apresentado na Figura 9, Hofmann estima que 28% do total de recursos do projeto é alocado para as atividades de ER. Além disso, a equipe costuma ser distribuída de maneira balanceada: 11% do esforço de projeto para elicitacão, 10% para modelagem e 7% para validação e verificação. Para facilitar seu trabalho, normalmente as equipes utilizam modelos e exemplos de documentos de especificação elaborados em projetos anteriores (Hofmann 2001).

Balanceamento da Equipe de ER

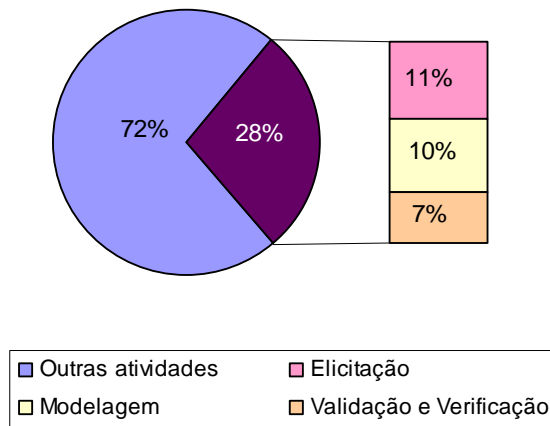


Figura 9 - Balanceamento da Equipe de ER (Hofmann 2001)

Por fim, Hofmann (Hofmann 2001) destaca que a orientação dos projetos de sucesso são dadas por requisitos priorizados pelos envolvidos. Isto permite que a equipe decida quais requisitos devem ser investigados, quando e com que nível de detalhe. Além disso, é indispensável manter a matriz de rastreabilidade dos requisitos. Este artefato permite o rastreamento de um requisito de sua origem até sua implementação. Ademais, equipes de sucesso validam e verificam, repetidamente, seus requisitos com múltiplos envolvidos, utilizando técnicas como revisão por pares, cenários e “walkthroughs” (Hofmann 2001).

4.1.4 Processo de ER Tradicional

Não existe um processo único que seja bom para todas as organizações. Cada organização deve desenvolver o seu próprio processo, que deve ser apropriado para o tipo de sistema que desenvolve, sua cultura organizacional, o nível de experiência e habilidade das pessoas envolvidas na ER. Existem várias formas de se organizar o processo de ER e eles não podem ser transferidos de uma empresa para outra. Para definir um bom processo de ER as organizações precisam abarcar pessoas que estejam realmente envolvidas na ER, e, caso julguem necessários, podem solicitar ajuda de consultores externos a fim de obterem uma perspectiva mais objetiva do que aquela dos envolvidos no processo (Sommerville 1998).

Poucas organizações têm um processo de requisitos explicitamente definido e padronizado. Muitos dos padrões de ER desenvolvidos até o momento envolvem apenas

saídas de processo, como estrutura de documentos de requisitos. Uma descrição do processo completo deve incluir quais atividades são realizadas, a estrutura ou agendamento destas atividades, quem é responsável por atividade, as entradas e saídas destas atividades e as ferramentas utilizadas para suportar a ER (Sommerville 1998).

De maneira geral, o processo de ER é realizado por meio da execução de sete funções distintas: concepção, levantamento, elaboração, negociação, especificação, validação e gestão. É importante notar que algumas dessas funções de ER ocorrem em paralelo e que todas são adaptadas às necessidades do projeto. Todas tentam definir o que o cliente deseja e servem para estabelecer uma fundação sólida para o projeto e a construção do produto que o cliente receberá (Pressman 2006).

No RUP a disciplina que descreve o processo de ER é designada apenas como “Requisitos”. O fluxo de trabalho desta disciplina (Figura 10) exhibe as chamadas habilidades-chaves necessárias para a execução eficiente do gerenciamento de requisito, sendo elas: Analisar o Problema, Compreender as Necessidades do Envolvidos, Definir o Sistema, Gerenciar o Escopo do Sistema, Refinar a Definição do Sistema e Gerenciar Requisitos Mutáveis. O fluxo de trabalho é exibido em ordem lógica e seqüencial, embora seja aplicado em ordem diferente, conforme a necessidade no decorrer do projeto.

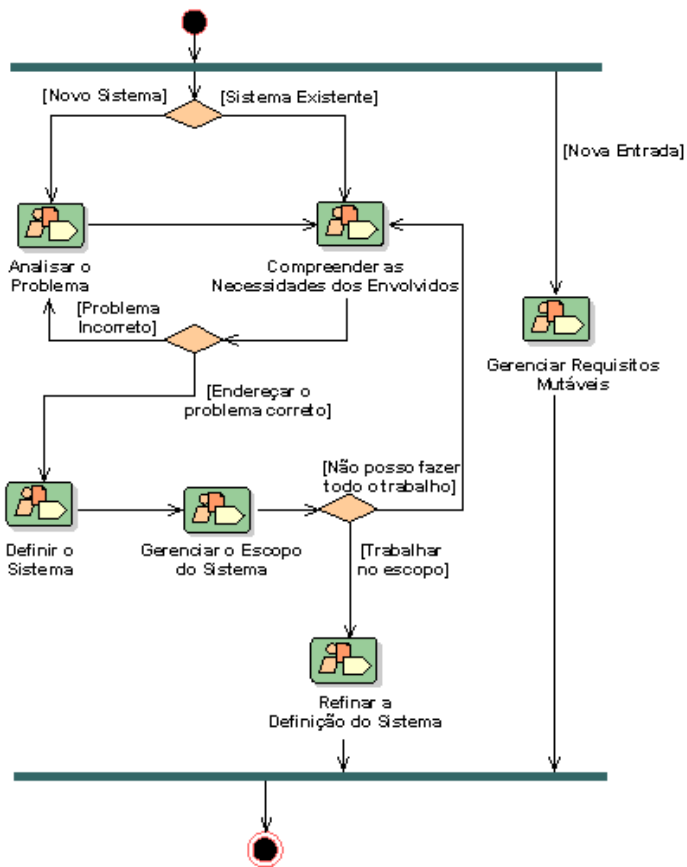


Figura 10 - Fluxo de Trabalho da Disciplina de Requisitos (RUP 2001)

4.1.5 Gerência de Requisitos

A Gerência de Requisitos consiste no processo de gerenciar mudanças nos requisitos do sistema. Afinal, os requisitos de um sistema sempre mudam para refletir as necessidades de mudanças solicitadas pelos envolvidos, as mudanças do negócio no qual o sistema deverá ser instalado, mudanças nas leis e regulamentações, etc (Sommerville 1998). Estas mudanças devem ser gerenciadas para garantir que estejam dentro do orçamento e do prazo estipulados.

As principais atividades da gerência de requisitos são o controle da mudança e a avaliação do impacto da mudança. Neste sentido, a gerência de requisitos exige que a rastreabilidade dos requisitos seja armazenada (Sommerville 1998). Ou seja, ligações entre os requisitos, as fontes de requisitos e o projeto do sistema devem ser rastreadas e guardadas, de maneira que estejam disponíveis no momento de avaliar o impacto e controlar as mudanças do sistema.

4.1.6 Envolvidos no Processo de Engenharia de Requisitos

Os envolvidos no processo de ER são as pessoas ou organizações afetadas pelo sistema e que têm uma influência direta ou indireta nos requisitos deste sistema (Sommerville 1998). Os envolvidos incluem usuários finais do sistema, gerentes e outros envolvidos no processo organizacional influenciados pelo sistema, engenheiros responsáveis pelo desenvolvimento e manutenção do sistema, clientes, etc.

4.1.7 Revisões de Requisitos

Revisões dos requisitos configuram-se como a técnica mais utilizada na validação de requisitos (Sommerville 1998). Elas incluem um grupo de pessoas que lêem e analisam os requisitos em busca de possíveis problemas. Algumas das técnicas utilizadas para este fim são: a inspeção, o walkthrough e a revisão por pares.

A inspeção consiste na leitura passo a passo do documento, a revisão é feita confrontando-o com uma lista de itens que define o escopo da revisão (lista de verificação). A equipe de inspeção do projeto geralmente é composta de quatro pessoas sendo que uma delas atua no papel de moderador. Já o walkthrough consiste na leitura passo a passo do documento revisado em uma reunião formal na qual o profissional que elaborou o documento participa. A principal diferença dos dois métodos é a maneira como a revisão é conduzida, enquanto a inspeção é baseada em uma lista de verificação o walkthrough simula a execução do programa (Dórias 2001).

Outra técnica de revisão é a revisão por pares. Neste método duas pessoas revisam o produto com o objetivo de discutir pontos críticos a fim de obter benefícios dos pontos de vista divergentes (Dórias 2001).

4.2 Requisitos em Sistemas Embarcados

Sistemas embarcados possuem características bem específicas no que diz respeito a seus requisitos. Nas seções seguintes são abordados alguns destes aspectos.

4.2.1 Requisitos Não Funcionais

Conforme já foi citado no capítulo 2, este tipo de sistema deve obedecer a restrições severas no que diz respeito a requisitos temporais, de peso, tamanho, mobilidade, segurança, confiabilidade, consumo de energia e memória. Nota-se, portanto, que um dos maiores desafios no desenvolvimento de sistemas embarcados é o atendimento de seus

requisitos não-funcionais, ou seja, aqueles cuja preocupação não é especificamente com a funcionalidade do sistema (Sommerville 1998).

No entanto, os requisitos não funcionais citados anteriormente não se configuram como os únicos complicadores do domínio de sistemas embarcados. Um estudo publicado por (Nars 2002) destaca, ainda, que softwares embarcados possuem propriedades como:

1. Estarem, com freqüência, embutidos em sistemas muito maiores e complexos;
2. Estarem acoplados a um hardware, de maneira que sensores e atuadores são os responsáveis por controlar as atividades;
3. Normalmente, a parte física do sistema já está estabelecida, devendo-se adaptar o software ao hardware existente;

Existe, ainda, a demanda do mercado por produtos de baixo custo e curto prazo de desenvolvimento (Yamaura 2003). Ou seja, estes softwares, além de possuírem características extremamente complexas, devem garantir uma rápida entrega e produtos de altíssima qualidade (S. K. Lee 2007). Ademais, muitos destes produtos são dirigidos à inovação, como no caso de celulares, carros e aviões. E neste caso, ser o primeiro a lançar um produto no mercado garante um lucro superior. Mas isto somente é possível caso o retorno do investimento exceda os custos de desenvolvimento e produção (Puschig 2004).

Aliado a estas características têm-se o fato de que os requisitos não funcionais são geralmente subjetivos, uma vez que podem ser vistos, interpretados e conceituados de forma diferente por diferentes pessoas. Apesar dos requisitos funcionais também serem afetados por pontos de vista variados, nos não funcionais este problema é potencializado. Isto ocorre devido ao fato dos requisitos não funcionais serem, por natureza, mais abstratos, além de normalmente serem especificados de maneira mais breve e vaga (Cysneiros 2001). Ademais, requisitos não funcionais freqüentemente interagem entre si, já que na tentativa de satisfazer um requisito pode-se prejudicar ou ajudar a satisfazer outros. Lidar com os requisitos não-funcionais não é uma tarefa fácil, mas seu tratamento é vital para a construção de sistemas embarcados.

Outro aspecto importante no que diz respeito a requisitos não funcionais é o fato de serem descobertos ou tratados tardiamente no projeto. E um dos motivos dos requisitos não-funcionais não serem definidos e analisados logo no início do projeto é o fato de não serem cobertos pela maioria dos métodos de Engenharia de Requisitos (Sommerville 1998).

A explicação para isto é a dificuldade para definir tal tipo de requisitos, e Sommerville (Sommerville 1998) apontou algumas destas causas:

1. Algumas restrições somente são descobertas na fase de projeto;
2. Certas restrições são altamente subjetivas, sendo determinadas, apenas, por avaliações empíricas;
3. Os requisitos não-funcionais tendem a estar relacionados com mais de um requisito funcional, dificultando a explicitação desta dependência;
4. Os requisitos não-funcionais tendem a conflitar e contradizer uns aos outros;
5. De uma maneira geral, não existem regras definidas para expressar os requisitos não-funcionais.

Buscando resolver estes problemas, Gilb (Gilb 1989) sugere a utilização de um método para especificação de requisitos não-funcionais denominado PLanguage. A PLanguage ou “Planning Language” (Gilb 1989) permite a medição e o teste da qualidade dos requisitos não-funcionais especificados e possui benefícios como facilidade de aprendizado e flexibilidade, além de ser compacta e prevenir omissões através de um conjunto consistente de parâmetros de qualidade que podem ser utilizados nas especificações (Gastaldo 2003). Sua forma de apresentação é composta por um conjunto de palavras-chave nas quais os requisitos devem ser especificados. Seu formato, atributos e conceitos são apresentados na Tabela 4 (Gastaldo 2003).

Tabela 4 – Definição das palavras-chave da linguagem de especificação PLanguage (Gastaldo 2003)

Palavras-Chave	Descrição
Tipo	Etiqueta, rótulo, identificador persistente e único do requisito
Descrição	Descrição simples e breve do conceito principal ou significado geral do requisito
Stakeholder	Envolvidos/Afetados pelo requisito
Escala	Escala usada para quantificar o requisito
Métrica	Processo ou método para medir escalas dos requisitos
Método	Método para medir a escala
Freqüência	Freqüência para medição
Responsável	Pessoas/Departamento responsável por fazer as medições

Registro	Onde/Quando as medidas devem ser reportadas
Nível Mínimo	O nível mínimo requerido para evitar falhas
Plano	Nível para obter sucesso exigido
Nível Sucesso	Como prolongar, aumentar, alongar o sucesso
Nível Desejado	Nível desejável de sucesso que não pode ser atingido através dos métodos atuais
Histórico	Resultados anteriores para comparação (histórico)
Tendência	Tendência histórica
Histórico de Sucesso	O melhor resultado obtido
Definição	Definição oficial do termo
Autoridade	Pessoa, grupo ou nível de autorização

Como pode ser visto na Tabela 4, o formato da PLanguage é claro para qualquer integrante do projeto. A utilização deste método tem como objetivo resolver os problemas de falta de técnicas para o levantamento de requisitos, falta de detalhamento e clareza nas especificações, confusão entre requisitos funcionais e não-funcionais e a falta de formatação e apresentação. Além disso, o PLanguage ainda tem a vantagem de os testes poderem ser feitos com base na própria especificação, eliminando o trabalho da elaboração de um documento de especificação e outro de testes (Gastaldo 2003).

4.2.2 Requisitos de Hardware x Requisitos de Software

Outro ponto importante, no que se refere aos requisitos em sistemas embarcados, é a difícil separação, durante a modelagem de requisitos, entre o hardware e o software, uma vez que estão profundamente acoplados e são altamente interativos (Nars 2002). Os softwares de sistemas embarcados normalmente controlam componentes de hardware que estão dentro dos limites do próprio sistema em desenvolvimento (Nars 2002).

4.2.3 Invenção de Requisitos

É comum, na Engenharia de Software, a assertiva de que os fornecedores de requisitos muitas vezes não sabem ou não conseguem expressar o que esperam do sistema. Em sistemas embarcados esta é uma realidade com a qual os engenheiros de software devem lidar com frequência ainda maior, visto que são responsáveis pela construção de celulares, eletrodomésticos, carros e aviões de última geração. E criar novos

produtos requer a invenção de novos requisitos, requisitos estes que nem mesmo os envolvidos sabem ainda quais são. É papel dos engenheiros de requisitos “inventar alguma coisa melhor” (Robertson, Eureka! Why Analysts Should Invent Requirements 2002). Isto exige, além de criatividade, um profundo conhecimento, por parte da equipe, a cerca do domínio do negócio (Puschnig 2004).

4.3 Processo de ER para Sistemas Embarcados

Se, conforme apontado por (Sommerville 1998) poucas organizações têm um processo de requisitos explicitamente definido e padronizado, processos de ER específicos para sistemas embarcados são ainda mais difíceis de encontrar. A literatura sobre o assunto é restrita, de maneira que não foi possível encontrar sequer uma fonte que definisse um processo completo de ER para sistemas embarcados. O único estudo que se aproximou do procurado foi um inventário publicado por (Graaf 2003).

Diante desta dificuldade a presente seção foi dividida de acordo com os assuntos abordados nas diversas fontes consultadas. Tendo sido descritas, em cada tópico, as práticas mais comuns utilizadas durante o processo de ER para softwares embarcados.

4.3.1 Contexto das Empresas Desenvolvedoras de SE

Um estudo bastante abrangente em relação ao desenvolvimento de softwares embarcados foi publicado por Graaf (Graaf 2003). De acordo com este inventário realizado em três países europeus, envolvendo sete companhias e um instituto de pesquisa, a maioria das companhias que desenvolve softwares embarcados não os vende. Eles vendem, na realidade, telefones celulares, CD players e outros produtos. O software nestes produtos constitui apenas uma parte, certamente importante, do produto como um todo. No entanto, quanto mais transparente e acoplado ao hardware ele for, mais invisível ele será do ponto de vista do produto (Henzinger 2007). Na realidade, os softwares somente são visíveis através dos aprimoramentos nas funções e na performance dos produtos (Henzinger 2007).

Além disso, grande parte das empresas possui um desenvolvimento dirigido ao hardware, ou seja, apenas quando o hardware já está em um estágio avançado de desenvolvimento é que o software começa a ser desenvolvido (Graaf 2003). Isto significa que o software deve, necessariamente, adaptar-se ao hardware construído. Este aspecto revelou-se um problema, na medida em que muitas dificuldades que poderiam ser resolvidas no domínio do hardware acabam solucionadas pelo software (Graaf 2003). E esta

situação pode ser agravada com a frequência cada vez maior do modelo de negócios horizontal, onde a responsabilidade da produção de um novo projeto é distribuída entre companhias distintas (Sangiovanni-Vincentelli 2004). O software assume, assim, o papel de integrador dos diversos componentes.

Outra consequência, observada por (Graaf 2003), deste desenvolvimento orientado ao hardware é que em algumas companhias os arquitetos de software não eram sequer envolvidos nas decisões de projeto em nível de sistema. Os desenvolvedores de softwares embarcados sentiram que isto estava se tornando um problema sério e, em função disto, muitas companhias estão mudando, de maneira que os arquitetos de software estão cada dia mais envolvidos nas decisões em nível de sistema.

Dado o contexto em os softwares embarcados estão inseridos, não é de se surpreender que o ele seja, hoje, a parte mais cara e menos confiável das aplicações de sistemas embarcados (Henzinger 2007).

4.3.2 Processos de Desenvolvimento de SE Sob o Ponto de Vista da ER

De acordo com as pesquisas de (Graaf 2003), o modelo mais comum de processo de desenvolvimento de sistemas embarcados utiliza uma abordagem *top-down*. Ou seja, nestes modelos as refinações de requisitos e arquitetura são realizadas em várias iterações subseqüentes, partindo do nível de sistemas até chegar ao nível de componente. A Figura 11 apresenta este modelo.

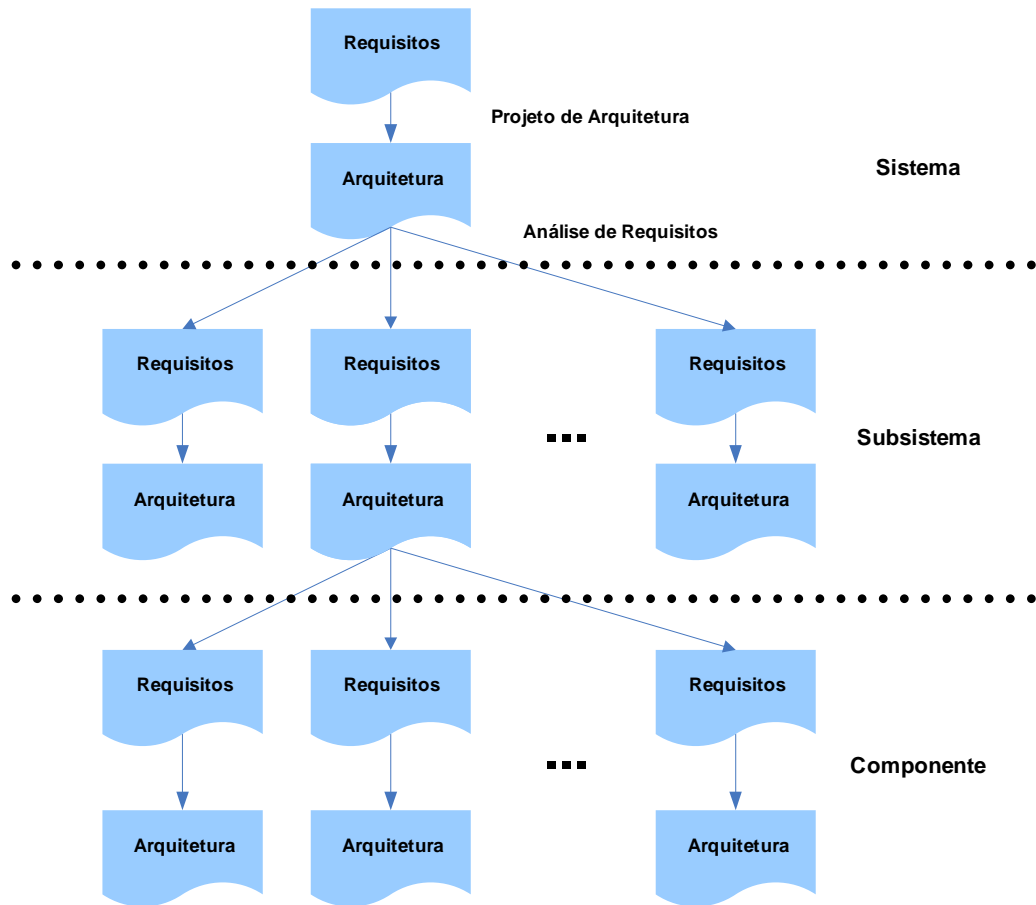


Figura 11 - Decomposição do processo de desenvolvimento de sistemas embarcados (Graaf 2003)

Já a metodologia de especificação de requisitos elaborada por (Lattemann 1997) consiste de cinco passos. Em cada passo as características relevantes do sistema para aquele passo são adicionadas à especificação de requisitos, conforme representado na Figura 12.

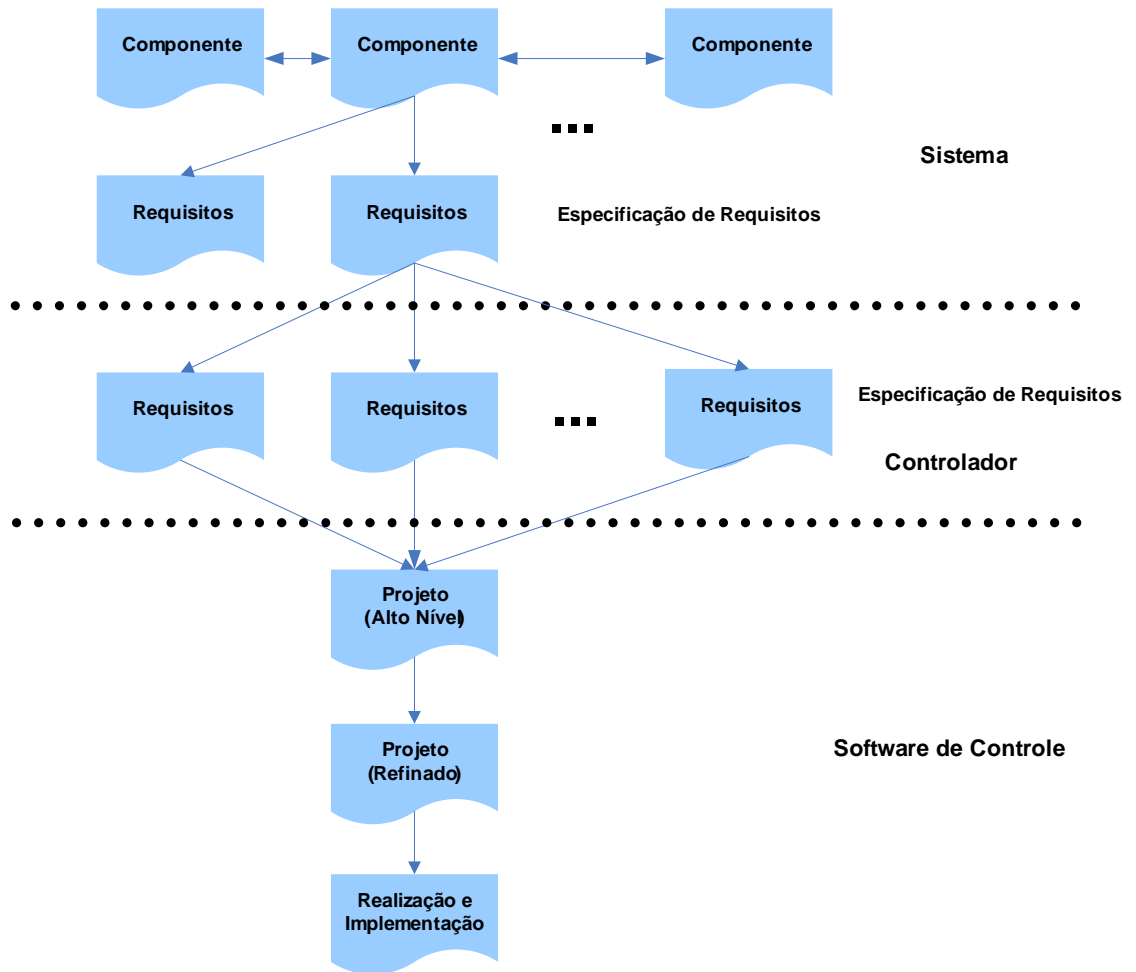


Figura 12 - Processo de Especificação de Requisitos (Lattermann 1997)

Neste processo (Lattermann 1997) defende que primeiro devem ser descritos os requisitos do sistema como um todo, sem considerar os possíveis erros e falhas. Para facilitar esta etapa (Lattermann 1997) sugere que o sistema seja dividido em componentes e que sejam criadas conexões entre estes. A próxima etapa é a especificação dos requisitos do controlador, definindo-se as interfaces entre hardware e software. Depois é construído o projeto em alto nível e, por fim, este projeto é refinado, podendo-se partir para a realização e implementação.

Neste ponto, a abordagem de especificação de requisitos elaborada por (Lattermann 1997) não se distancia muito do processo genérico desenhado por (Graaf 2003). Pois, apesar de não utilizarem os mesmos termos, em ambos o sistema é dividido em sistemas menores e a especificação é refinada em etapas.

Por outro lado o processo de elicitação e especificação de requisitos utilizado por (Nars 2002) toma como base a técnica de caso de uso e possui 7 etapas principais, conforme ilustrado na Figura 13.

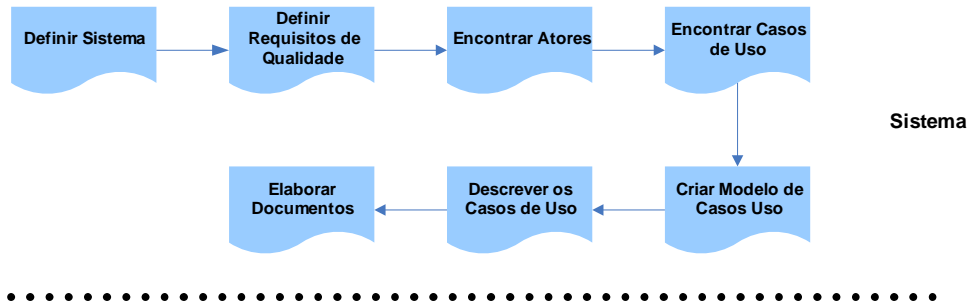


Figura 13 - Interação indireta entre o piloto e um caso de uso do TRCS (Nars 2002)

O primeiro passo desta abordagem é a definição do sistema, que tem por objetivo restringir o escopo e entender o sistema a ser construído. A segunda etapa consiste na especificação dos requisitos de qualidade, onde devem ser definidas explicitamente as restrições do sistema. Daqui em diante o processo de (Nars 2002) muito se assemelha aos processos tradicionais de ER: são encontrados os atores, os casos de uso, é construído o modelo de caso de uso e são elaborados os documentos do sistema.

O diferencial desta abordagem é o fato de (Nars 2002) defender para encontrar os casos de uso o adequado é trabalhar baseado em funções e não em atores. Além de proporcionar maior flexibilidade, esta técnica permite que se encontre com maior habilidade todos os casos de uso funcionais do sistema. Nars (Nars 2002) ressalta ainda que muito cuidado deve ser tomado a fim de identificar apenas o que é necessário para o sistema em especificação, pois, no domínio de softwares de controle de sistemas embarcados é um grande desafio levantar apenas as funcionalidades do sistema a ser construído, dada a alta interatividade entre ele e os demais sistemas envolvidos. Além disso, (Nars 2002) aconselha que seja utilizado um processo iterativo e progressivo de refinamento.

Como pode ser visto, diferentemente das abordagens de (Lattemann 1997) e (Graaf 2003), Nars (Nars 2002) não divide a especificação de requisitos em subsistemas ou separa as especificações de hardware e software. Pelo contrário, Nars defende a especificação do sistema como todo, dada a difícil separação de hardware e software no contexto de sistemas embarcados (Nars 2002).

4.3.3 Metodologias para Especificação dos Requisitos em SE

De acordo com (Graaf 2003) a UML ainda não é uma prática comum, mas muitas empresas já têm considerado sua possibilidade de aplicação na ER. Mesmo não tendo adotado a UML como padrão, muitos projetos utilizaram diagramas para representar requisitos cujas formas, em sua maioria, lembravam o estilo de diagrama da UML ou outras notações.

Ainda segundo (Graaf 2003), as construções mais utilizadas pelas companhias foram os casos de uso. No entanto, alguns projetos chegaram a usar diagramas de seqüência para realizar os casos de uso, outros aplicaram diagrama de classes para modelar o domínio. Mas em muitos casos a interpretação das notações UML não estava muito clara.

Neste aspecto, (Puschnig 2004) defendem a utilização da técnica de caso de uso, pois de acordo com suas pesquisas, a modelagem por casos de uso foi de grande valia para o entendimento inicial do sistema, por parte dos especialistas, mesmo quando estes modelos ainda eram incompletos ou incorretos, uma vez que serviam como base para as discussões.

Apesar dos inúmeros benefícios apontados pela literatura em relação à utilização de casos de uso, (Nars 2002) ressalta que a aplicação desta técnica em um caso real da indústria de sistemas embarcados revelou certa confusão pelas seguintes razões: a técnica de caso de uso é carente de definições apropriadas para a adaptação a sistemas embarcados, falta uma clara definição do processo de aplicação prática da técnica de caso de uso e as relações entre requisitos e casos de uso não claras o suficiente.

Buscando resolver estas questões, (Nars 2002) não apenas propôs, mas aplicou em um sistema do ramo aviônico, algumas adaptações na técnica de caso de uso a fim de adequá-la ao universo de sistemas embarcados, conforme descrito na seção 2.2.2

No entanto, antes de definir o processo de elicitação e especificação de requisitos baseado em casos de uso, (Nars 2002) esclarece como são interpretados os principais construtores da técnica de caso de uso ao adaptar esta técnica ao domínio de sistemas embarcados. Tomando como base os principais construtores da técnica de caso de uso Figura 14 o autor destaca que:



Figura 14 - Os principais construtores da técnica de caso de uso (Nars 2002)

- É mais apropriado considerar como “sistema” a composição de ambos, hardware e software, visto que é muito difícil separá-los durante a modelagem de requisitos;
- Um ator representa o papel de uma entidade externa e não a própria entidade externa, ressaltando-se que uma entidade pode ser qualquer coisa, humana ou não (outros sistemas, software ou hardware, uma tempestade, um pássaro, etc.);
- Um ator está situado fora da fronteira do sistema a ser modelado;
- Um ator deve interagir diretamente com o sistema a ser modelado;
- Um caso de uso pode ser iniciado internamente pelo sistema, não sendo necessária a intervenção de um ator. Esta adaptação visa atender à peculiaridade de que em softwares de controle de sistemas embarcados a maioria das funções de controle é realizada sem que haja uma entrada externa;
- Um caso de uso pode descrever funcionalidades internas de um sistema e não apenas seu comportamento externo;
- Muitas vezes durante a elicitação de requisitos os autores sentiram a necessidade de modelar uma interação temporária entre um ator e um caso de uso. Em um avião, por exemplo, o piloto é o ator responsável por controlar todo o hardware do avião, no entanto, caso se esteja modelando apenas o sistema de controle do reverso (designado TRCS) do avião, o piloto realizará a interação através de outros sistemas do avião. Para resolver esta questão foi criada a ligação indireta entre caso de uso e ator, conforme pode ser observado na Figura 15.

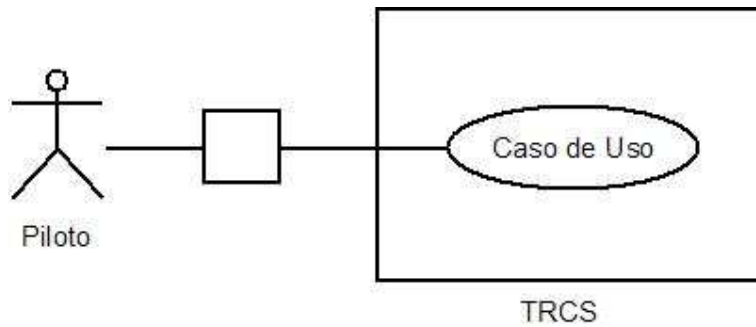


Figura 15 - Interação indireta entre o piloto e um caso de uso do TRCS (Nars 2002)

4.3.4 Envolvidos no Processo de ER para SE

Outro complicador comum no desenvolvimento de sistemas embarcados é a participação de muitos envolvidos, e isto é ainda mais notório durante a ER. A

Figura 16 apresenta os envolvidos encontrados com mais freqüência no processo de ER de acordo com o inventário de Graaf (Graaf 2003). Segundo (Graaf 2003), na primeira fase da ER o cliente determina os requisitos funcionais e não funcionais, e, dependendo do domínio do produto, estes requisitos são negociados entre o cliente e as áreas de marketing e venda ou mesmo diretamente com os desenvolvedores. Ou seja, o Engenheiro de Requisitos não aparece no momento da negociação.

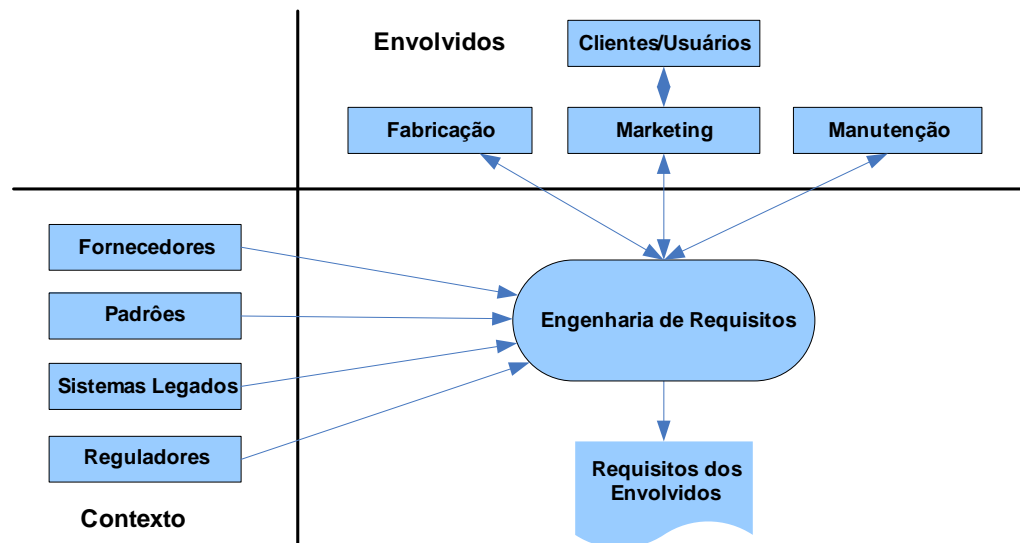


Figura 16 - Envolvidos do desenvolvimento de sistemas embarcados (Graaf 2003)

Um estudo realizado por (Puschig 2004) identificou que, para o sucesso da ER do domínio de SEs, além do núcleo da equipe de desenvolvimento formada por engenheiros de requisitos, analista de processo, programadores, etc., é necessário o envolvimento de

especialistas de diferentes domínios. Isto se faz necessário em virtude do pouco conhecimento que os envolvidos muitas vezes possuem acerca do sistema a ser desenvolvido. A solução encontrada, portanto, foi o envolvimento de especialistas nas mais diversas áreas.

Em sua maioria, os especialistas requisitados estavam associados a uma área específica de conhecimento: desenvolvedores, professores, gerentes, engenheiros de produção, consultores e fornecedores. Os engenheiros de software e os especialistas trabalhavam em conjunto, de maneira que os primeiros possuíam uma concepção do sistema como um todo, ao passo que os segundos tinham o conhecimento detalhado de um ou mais aspectos técnicos. No entanto para definir qual especialista será necessário e em que momento deverá ser requisitado é necessário, em primeiro lugar, estabelecer as fronteiras do sistema.

Com tantas pessoas envolvidas no projeto fica complicado gerenciar os diferentes requisitos provenientes de todas estas fontes. Esta característica é marcante, especialmente, em projetos grandes (Graaf 2003). Ademais, normalmente muitos dos especialistas requisitados não estão situados em um mesmo local e nem estão disponíveis a todo o momento, visto que podem estar alocados em mais de um projeto. Portanto, é papel do gerente de projeto prover os arranjos necessários para garantir o efetivo e eficiente envolvimento dos membros do projeto (Puschnig 2004).

4.3.5 Modelos de Documentos para Especificação dos Requisitos em SE

Segundo (Graaf 2003), as companhias estudadas normalmente especificavam seus requisitos em linguagem natural, registrando as informações em processadores de texto. Normalmente utilizam modelos para estruturar os documentos, assim teriam um guia do que devia ser levantado. No entanto, nem todos os projetos de uma mesma companhia utilizavam os modelos, de maneira que podiam ser encontrados diferentes modelos de documentos dentro de um mesmo projeto.

Como nos softwares embarcados as propriedades não funcionais são, geralmente, de grande importância, a expectativa era de que os modelos desenvolvidos pelas empresas estudadas por (Graaf 2003) reservassem uma seção apenas para requisitos não funcionais, mas não foi o que se observou. Alguns projetos ainda chegaram a especificar características como restrições de tempo-real em documentos separados, mas de maneira geral, estes e outros requisitos como consumo de energia e memória não eram explicitamente especificados e projetados.

4.3.6 Gerenciamento de Requisitos em SE

Outro complicador destacado por (Graaf 2003) é o fato de muitas companhias construírem projetos em cima de projetos anteriores. Ou seja, estes projetos reutilizavam as especificações de requisitos, mesmo para o desenvolvimento de uma nova linha de produto. Conseqüentemente, deixar a documentação de requisitos consistente é uma tarefa árdua, pois para deixar todos os produtos e documentos do desenvolvimento consistentes, é preciso analisar precisamente o impacto de novas características. No entanto, os projetos estudados freqüentemente não explicitavam as relações entre requisitos, então a análise de impactos era muito difícil. Rastrear os requisitos era difícil por que as relações (por exemplo, entre requisitos e componentes arquiteturais) eram muito complexas para serem especificadas manualmente e as ferramentas de gerenciamento de requisitos disponíveis não pareciam resolver este problema, apesar de algumas versões customizadas funcionarem em alguns casos.

Esta rastreabilidade é um aspecto essencial do gerenciamento de requisitos, visto que entender os riscos associados ao impacto do software em seu ambiente é fundamental para assegurar a dependabilidade e a segurança no funcionamento de sistemas embarcados. Infelizmente, a análise de risco de sistemas embarcados tem sido negligenciada e apenas recentemente tem recebido atenção da comunidade de engenharia de software (Hewett 2005).

4.3.7 Outros Aspectos Importantes no Processo de ER para SE

O estudo realizado por (Puschig 2004) em dois projetos da Daimler Chrysler revelou outros dois aspectos, ainda não abordados até o momento, que se mostraram de grande importância para o sucesso da ER no domínio de sistemas embarcados:

- **Orientação a objetivos:** abordagens orientadas a objetivos são comuns na literatura, mas em sistemas altamente inovadores como os embarcados esta abordagem é essencial para descrever apropriadamente o sistema. Tão importante quanto a orientação a objetivos é a utilização de camadas de abstração, visto que é muito difícil solicitar aos especialistas que discorram sobre suas ideais de maneira mais abstrata.
- **Revisões:** as revisões revelaram-se uma maneira de completar a especificação de requisitos e de elevar os requisitos a um nível confiável e maduro. Uma conseqüência das revisões, no entanto, é a quantidade de

tempo gasta em cada uma, visto que segundo suas pesquisas, os autores chegaram a despende um dia inteiro ou até mais em cada revisão. Entretanto, as revisões aparecem como uma oportunidade para capturar e consolidar os requisitos com a ajuda dos especialistas, ou seja, é mais uma chance de realizar um trabalho efetivo e de exercitar a sinergia.

(Puschnig 2004) sugerem que durante as revisões sejam executadas as seguintes tarefas: coleta dos comentários gerais sobre a compreensão, estruturação e apresentação do documento de especificação inspecionado. Estes comentários devem ser consolidados e de acordo com a relevância deve ser feito um *ranking*, de maneira que possam ser negociados e documentados. Como saída desta etapa deve ser gerado um relatório de revisão com os comentários e a concordância de todos os envolvidos.

4.4 Considerações Finais

Conforme exposto neste capítulo, sistemas embarcados não possuem processos definidos e institucionalizados para ER. Além disso, as abordagens para a levantamento e especificação de requisitos apresentadas mostraram-se inadequadas para o contexto deste tipo de sistema. De maneira que estes processos não conseguiram resolver ou minimizar as dificuldades no desenvolvimento de softwares embarcados.

O capítulo a seguir apresentará a abordagem GQM. Este método será utilizado na elaboração dos objetivos, questões e métricas da avaliação a ser submetida às empresas participantes do programa de medições. A geração dos indicadores a partir das informações coletadas nesta pesquisa permitirá a criação de um processo que melhor se adapte à realidade das MPEs desenvolvedoras de softwares embarcados.

5. ABORDAGEM GQM

Conforme descrito no Capítulo 03, o aumento dos custos e da complexidade de desenvolvimento de sistemas aliados a necessidade de ampliação da competitividade das organizações, tem levado a uma exigência cada vez maior pela qualidade dos softwares produzidos (Farias 2006)

Além da dificuldade de se desenvolver software com qualidade a identificação dos problemas que ocorrem durante o desenvolvimento do produto de baixa qualidade é um ponto extremamente custoso para as empresas. Para melhorar a qualidade dos sistemas desenvolvidos é necessário entender os recursos, processos e os produtos e seus impactos na organização. Isso requer que a empresa capture conhecimento do desenvolvimento de software (por meio da aplicação de programas de medições ou utilização de bases de conhecimento) e o aplique no planejamento, controle e melhoramento de seus processos e de seus projetos (Wangenheim e Ruhe 1999).

A abordagem GQM é um método de especificação de programas de medição de software que permite que as informações obtidas sejam utilizadas para acompanhamento do alcance dos objetivos traçados para o programa (Goethert e Fischer 2003).

Este capítulo aborda conceitos relacionados ao método GQM. A seção 5.1 descreve o modelo GQM e seus níveis, e identifica suas fases. As seções 5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.1.5 descrevem em detalhes as fases do modelo GQM, suas atividades e os produtos gerados. Por fim, a seção 5.2 descreve as considerações finais do capítulo.

5.1 O Método Goal Question Metric (GQM)

A metodologia GQM foi originalmente concebida durante as décadas de 70 e 80 por Victor Basili e David M. Weiss, posteriormente foi estendida e formalizada por Rombach e sua equipe (Basili e Rombach 1994). Proposta com o objetivo de avaliar defeitos em projetos no Centro Espacial da NASA, a metodologia GQM tem expandido sua utilização para um grande contexto de aplicações na área de Engenharia de Software, principalmente por se adaptar facilmente para a avaliação de qualquer tipo de problema (Basili e Rombach 1994) (Kopanas, Sylaidis e Nakakis 1997). Andrade e Souza, por exemplo, propõem a definição de um modelo de métricas que avalie processos de software (Anquetil, Oliveira e Souza 2005). Soligen e Bergouth (Berghout e Solligen 1999), apresentam estudos para utilização do GQM na definição de um modelo de métricas, além de várias outras propostas de utilização da metodologia (Anquetil, Oliveira e Souza 2005) (Basili e Rombach 1994).

O principal objetivo da metodologia GQM é fornecer e caracterizar um melhor entendimento dos processos, produtos, recursos e ambiente para estabelecer bases para comparações com trabalhos futuros ou até mesmo com metas (índices) de mercado (Anquetil, Oliveira e Souza 2005). Portanto, podemos defini-la como uma abordagem sistemática que define e integra objetivos a modelos de processo, produtos e serviços sob a perspectiva de qualidade baseada em necessidades específicas dos projetos e das organizações através de um programa de medições (Gladcheff, Sanches e da Silva 2001).

O GQM é amplamente citado na literatura como um mecanismo eficiente utilizado para planejar e definir objetivos da medição e avaliar produtos e processos de software, pois é uma abordagem de mensuração que é orientada pela avaliação orientada a objetivos (Gladcheff, Sanches e da Silva 2001) (Ramos 2004). “O princípio da metodologia GQM é a definição de um ou mais objetivos que servem de rota para a definição de questões que orientam a elaboração de métricas para avaliação dos objetivos especificados” (Anquetil, Oliveira e Souza 2005). O modelo GQM parte do princípio que a implementação de metas operacionais e mensuráveis para melhoria de software deve seguir uma abordagem top-down, no entanto a interpretação dos dados coletados durante a avaliação deve seguir a abordagem *bottom-up* (Wangenheim e Ruhe 1999). Portanto, a abordagem estabelece que primeiramente os objetivos devem ser traçados para que posteriormente sejam definidas suas questões e métricas para avaliação. Sendo assim, pode-se subdividir o processo em três níveis distintos (Basili e Rombach 1994):

- **Conceitual (Goal):** Definição do objetivo (produto, processo ou serviço) para uma variedade de questões, propósito da aplicação do GQM (ex.: melhorar, caracterizar ou controlar), foco de qualidade (características de qualidade), ponto de vista (interessados na aplicação do GQM) e ambiente (contexto para aplicação do resultado).
- **Operacional (Question):** Definição do conjunto de questões em linguagem natural propostas para caracterizar a forma de avaliação para um objetivo específico definido no nível conceitual.
- **Quantitativo (Metric):** Definição do conjunto de métricas associadas a cada questão que especificam em termos quantitativos e avaliáveis as informações que se deseja obter durante a avaliação.

Segundo (Anquetil, Oliveira e Souza 2005) apud (Berghout e Solligen 1999), a implantação do modelo GQM é realizada em quatro fases, que serão descritas nas seções a seguir. A Figura 17 ilustra as os dados produzidos em cada uma das fases do modelo GQM:

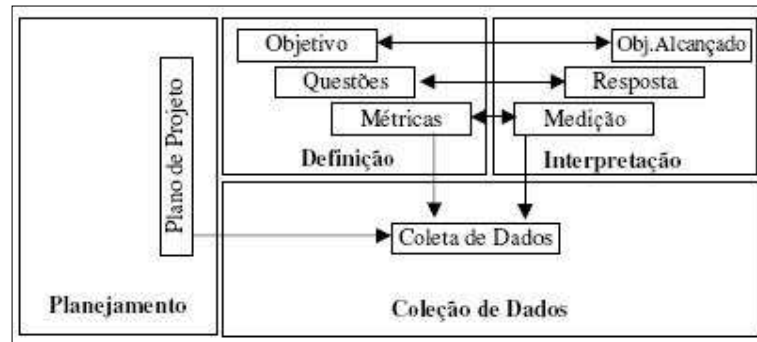


Figura 17 - Fases da Abordagem GQM (Berghout e Solligen 1999)

5.1.1 Planejamento

Nesta fase são realizados estudos e definições iniciais para introdução do programa de avaliação. Suas principais atividades são a coleta de todas as informações necessárias para o início da avaliação, a designação, o treinamento e a motivação da equipe que irá participar do GQM, a seleção da área a ser melhorada, a identificação dos projetos que irão participar da aplicação do método, a identificação de quando e como os dados serão coletados e a criação do Plano de Projeto. A fase de planejamento pode ser subdividida em quatro sub-fases (Laboratory s.d.):

4. Definição da equipe GQM: Alguns pontos devem ser levados em consideração quando a equipe é definida: a equipe deve ser capaz de definir objetivos e melhorias orientada pelo planejamento do projeto, o planejamento da coleta de dados deve ser elaborada e a equipe deve ser treinada para a interpretação dos dados.
5. Seleção da área de melhoria: Nessa fase diversas áreas processos ou produtos de software devem ser selecionadas. A escolha da área de melhoria deve ser realizada levando em consideração o negócio, os objetivos, os custos, o tempo os riscos e a qualidade da avaliação. Outros detalhes como problemas que podem ocorrer durante o processo de avaliação, influências externas, pessoas, processos e produtos envolvidos e conhecimento prévio das pessoas que estão

envolvidas no projeto sobre o processo de medição devem ser levados em conta na elaboração da lista de seleção.

6. Seleção do projeto e definição da equipe de projeto: A equipe GQM deve alinhar a equipe de projetos selecionada para a medição, com o objetivo de mantê-los cientes dos princípios do GQM enfatizando suas principais características.
7. Elaboração do Plano de Projeto: o Plano de Projeto é elaborado por meio de informações da equipe de projeto, o documento deve possuir:
 - o Programa de Medições: Breve descrição do Plano de Projetos.
 - o Introdução: Apresenta uma visão geral do programa de medições e contém uma descrição de como os objetivos de melhoria são traduzidos em objetivos de projetos de desenvolvimento de software.
 - o Programação (cronograma): Descrição completa das tarefas planejadas, quais formulários serão utilizados e que momento serão aplicados, resultados que devem ser obtidos e os custos e benefícios esperados.
 - o Organização: Define os objetivos da organização que são relevantes no programa de medições.
 - o Gerenciamento dos Processos: Contém os procedimentos de comunicação, as prioridades e descrições das atividades de controle de riscos.
 - o Treinamento e Promoção: Define como será o processo de treinamento e de divulgação do programa de medições.

5.1.2 Definição

Nesta etapa são definidos e documentados os objetivos, questões, métricas e hipóteses. A definição do programa de medições é composta das seguintes dimensões (Gladcheff, Sanches e da Silva 2001):

- Objeto de Estudo: O que será analisado.
- Objetivo: Envolve quatro aspectos – o propósito da análise (determinar, caracterizar, melhorar ou controlar alguma característica do objeto de medida), foco de qualidade (qual atributo do objeto será avaliado), ponto de vista (identifica

quais os interessados pelo resultado da pesquisa), e contexto (em qual ambiente está localizado).

- **Questões:** Conjunto de perguntas que expressam a forma de se obter informações em uma linguagem natural cujas respostas devem estar de acordo com os objetivos.
- **Métricas:** Conjunto de medidas que especificam em termos quantitativos e avaliáveis, as informações que se deseja obter durante as avaliações.

A fase de definição pode ser subdividida em nove sub-fases (Laboratory s.d.):

8. **Definição dos Objetivos da Medição:** Nesta fase os objetivos da medição devem ser formalmente definidos e estruturados. A Tabela 5 define algumas questões relevantes na definição dos objetivos de medição:

Tabela 5 – Questões de Definição dos Objetivos da Medição (Laboratory s.d.)

Qual?	o objeto que será avaliado.
Por quê?	entender, controlar e melhorar o objeto de medição.
Qual aspecto?	qual o foco da qualidade será avaliado no objeto de medição.
Quem?	as pessoas que avaliam o objeto de medição.
Contexto.	o ambiente no qual o objeto de medição será avaliado.

9. **Definição do Modelo de Processos de Software:** Nesta etapa podem ser utilizados modelos de processos de software existentes ou podem ser criados novos modelos. O modelo proposto deve ser completo e consistente de acordo com as definições do processo de medições.

10. **Entrevistas GQM:** A comunicação entre os perfis interessados na medição é essencial para o sucesso do programa. A extração de informação dos membros do projeto é comumente realizada, por meio de entrevistas individuais que podem ser guiadas por diagramas chamados *abstract sheets* que estruturam as informações a serem coletadas de forma que a equipe possua um ponto inicial para o refinamento das questões e métricas conforme descrito a seguir (Berghout e Solligen 1999) (Ramos 2004) e ilustrado na Tabela 6:

- Foco da qualidade: coleta e formalização das possíveis métricas a serem utilizadas na medição do objeto selecionado.
- Linhas de base das hipóteses: coleta e formalização dos envolvidos no programa métricas da sua opinião sobre os possíveis resultados de cada uma das métricas registradas (nesse caso é comum utilizar percentuais).
- Fatores de variação: coleta e formalização dos fatores que podem influenciar os resultados da medição.
- Impacto nas linhas de base das hipóteses: formalização do impacto e influência dos fatores de variação nos resultados das métricas registradas. Definição da dependência entre os fatores de variação e as métricas.

Tabela 6 - Abstract Sheets (Berghout e Solligen 1999)

Objeto	Propósito	Foco da qualidade e	Ponto de vista
<i>Foco da qualidade</i>		<i>Fatores de variação</i>	
<i>Linha de base</i>		<i>Impacto nas linhas de base</i>	

11. Definição de Questões e Hipóteses: As questões a serem definidas devem prover a correta interpretação dos objetivos relacionados. Para isso as questões devem ser definidas em um nível intermediário, não extremamente abstrato e nem extremamente detalhado. As hipóteses de resposta devem ser determinadas durante essa fase. Tanto as questões como as hipóteses de respostas devem ser examinadas e se necessário reformuladas para refletir os objetivos propostos.
12. Definição de Métricas: Nesta fase é necessário encontrar uma forma quantitativa de responder as questões e representar os objetivos definidos. Para a completa diminuição de erros as métricas elaboradas devem ser revisadas em relação a sua completude e consistência em relação do modelo da avaliação conforme exemplificado na Figura 18:

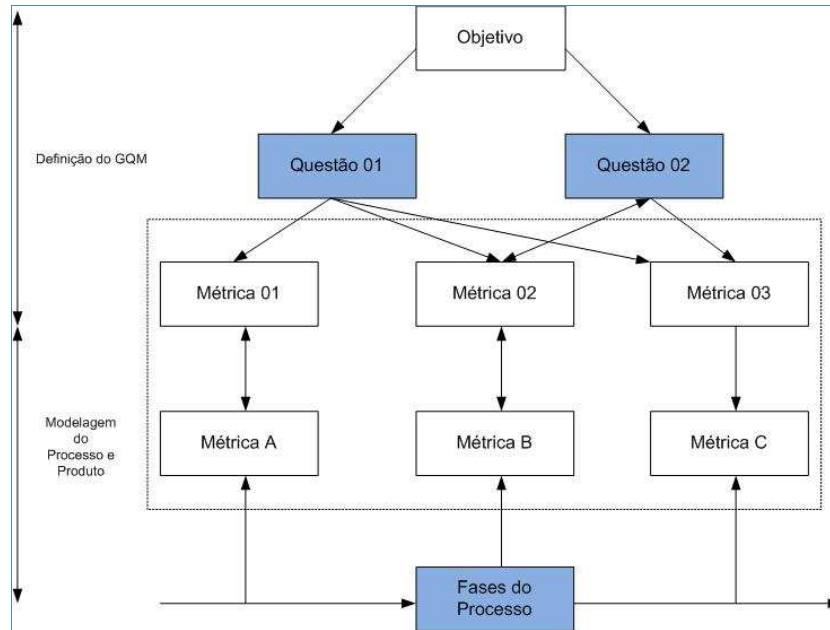


Figura 18 – Modelo de Avaliação GQM (Laboratory s.d.)

13. **Elaboração do Plano GQM:** Nesse documento são formalizados os objetivos, as métricas e as hipóteses do programa de medições definidos nas fases anteriores. O Plano GQM deve conter todas as informações necessárias para coleta e interpretação de dados.
14. **Elaboração do Plano de Medições:** Este documento é mais específico e refinado que o Plano GQM, ele deve conter definições formais e textuais das medições que serão efetuadas, como, por exemplo, quais as pessoas que irão realizar a coleta dos dados, ferramentas utilizadas e a data de realização das coletas. O Plano GQM também deve conter possíveis resultados da avaliação.
15. **Elaboração do Plano de Análise:** Este documento deve simular uma interpretação dos dados antes do início da medição. O Plano de Análise deve conter valores de métricas, gráficos e fluxogramas que estão de acordo com o Plano QGM.
16. **Revisão:** Nessa fase todos os documentos de aplicação do GQM devem ser revisados por todos os membros da equipe, e todos devem concordar com as definições e deliberações. Após essa fase a aplicação do GQM deve ser iniciada.

5.1.3 Coleta

Nesta fase são coletados os dados de avaliação, por meio do preenchimento dos questionários elaborados pela equipe GQM. Os dados da pesquisa são validados através da

verificação de não ocorrência de erros, consistência e completude dos questionários preenchidos. A fase de coleta pode ser subdividida em três sub-fases (Laboratory s.d.):

- Aguardar o Período de Prova: Antes da efetiva aplicação questionários, dos procedimentos e ferramentas de coleta de dados um período de proposição da medição deve ser executado. Para isso uma sessão de abertura de coleta de dados deve ser organizada, na qual os participantes da reunião (equipe de projeto) devem avaliar a aprovar formalmente os planos de medição, os formulários, as ferramentas e os procedimentos de coleta.
- Base de Métricas: Nessa fase os formulários de coleta de dados devem ser preenchidos pelos participantes da avaliação. A equipe GQM é responsável por acompanhar a coleta dos dados, agrupar e verificar a corretude dos dados coletados.
- Armazenamento dos dados coletados: Além das repostas coletadas nas avaliação, devem ser armazenadas as datas e atividades executadas por cada membro da equipe e o intervalo de tempo gasto para a sua execução.

5.1.4 Interpretação

Os dados obtidos na fase de coleta são analisados quantitativamente ou qualitativamente e conclusões são extraídas pela equipe de avaliação através da análise destes dados. Nesta fase as questões a serem avaliadas são respondidas com base na análise dos dados obtidos utilizando a abordagem *bottom-up*. Os pontos de melhoria são identificados a ações para melhoria são determinadas. A fase de interpretação pode ser subdividida em três sub-fases (Laboratory s.d.):

- Preparação das Sessões de Feedback: O Plano GQM contém todas as informações necessárias para a elaboração das sessões de feedback. A equipe GQM deve preparar todo o material da sessão de feedback que deve ser capaz de apoiar a análise e interpretação de dados, a definição da conclusão e a sua tradução em ações corretivas.
- Resultados das Medições: Após a sessão de feedback a equipe GQM deve elaborar um relatório da reunião contendo suas observações relevantes, conclusões e ações corretivas que foram formuladas durante as sessões de feedback.

- **Análise de Custo e Benefício do Programa de Medições:** A obtenção dos objetivos do GQM é um fator essencial para o programa de medições. Entretanto, avaliar os custos/benefícios estimados é importante do ponto de vista econômico e deve ser executado ao final da fase de interpretação.

5.1.5 Captura de Experiências

Os resultados da interpretação dos dados e as informações do próprio programa de medições são organizados em modelos e são armazenados em bases de experiência para sua disponibilização em projetos futuros.

5.2 Considerações Finais do Capítulo

Este capítulo apresentou conceitos relacionados ao método GQM. Foram detalhadas as fases de planejamento, descrição, coleta, interpretação e captura de experiências. Em cada uma das fases descritas foram identificadas as atividades executadas e os produtos gerados.

Como podemos perceber na descrição do capítulo, o modelo GQM é uma ferramenta que propicia a construção de programas que visam realizar a medição de qualquer objeto de interesse. Dessa forma o modelo será utilizado na avaliação das características do desenvolvimento de sistemas embarcados.

O capítulo 6 descreverá em detalhes a elaboração do programa de medições dos processos de desenvolvimento de sistemas embarcados de MPEs.

6. AVALIAÇÃO DO PROCESSO DE ENGENHARIA DE REQUISITOS PARA O DESENVOLVIMENTO DE SISTEMAS EMBARCADOS

Nos capítulos 02, 03 e 04 foram expostas as principais características dos sistemas embarcados, dos processos de desenvolvimento tradicionais e dos processos de desenvolvimento específicos para sistemas embarcados, além de processos de engenharia de requisitos para sistemas embarcados. O capítulo 5 apresentou a importância da utilização das métricas para a avaliação de produtos, processos e serviços de software, bem como a descrição de modelos de avaliação que podem ser usadas para melhorar e entender um software a ser avaliado.

Neste capítulo será apresentado o modelo de avaliação do processo de engenharia de requisitos baseado na abordagem Goal-Question-Metric (GQM). O objetivo da elaboração desse modelo é permitir o melhor entendimento do processo de engenharia de requisitos de sistemas embarcados (SE) em MPEs brasileiras, de forma a gerar indicadores que alimentem uma nova proposta de processo de Engenharia de requisitos para SE.

A organização do capítulo é realizada da seguinte forma: na seção 6.1.1 são apresentados o planejamento realizado para a seleção da metodologia de avaliação, a definição das equipes avaliadora e avaliadas, além da descrição do processo de comunicação e treinamento dos participantes. A seção 6.1.2 descreve o modelo de avaliação GQM do processo de engenharia de requisitos do desenvolvimento de SEs. A seção 6.1.4 descreve os procedimentos de medição, assim como a coleta dos questionários respondidos pelos participantes. As seções 6.1.5 e 6.1.6 apresenta a interpretação das informações coletadas e a captura de experiências do programa de medições. Finalmente, a seção 6.2 descreve as considerações finais do capítulo.

6.1 Aplicação do GQM

Apesar de existirem muitos paradigmas para a definição de objetivos e escopo de avaliações de software, optou-se pela utilização do GQM por ter se apresentado como um excelente mecanismo para o planejamento e definição de avaliações (Ramos 2004).

De acordo com o método GQM, para que algo possa ser medido de maneira eficaz é necessário, em primeiro lugar, traçar os objetivos da medição a fim de que sirvam como um guia para a criação das questões e, posteriormente, das métricas. Sendo assim, tomando-se como base a metodologia GQM, a avaliação foi dividida em quatro etapas – planejamento, definição, coleta e interpretação.

6.1.1 Planejamento

Na fase de planejamento são realizados estudos e definições iniciais para introdução do programa de avaliação. É nessa fase que se determina os objetivos de negócio de alto nível de interesse.

Com base nas definições do capítulo 5, a fase de planejamento do programa de medições foi iniciada com a realização de reuniões para discussão de questionamentos relacionados ao objetivo e ao foco da avaliação. Duas questões iniciais foram levantadas durante as discussões, conforme descrito na Tabela 7.

Tabela 7 - Questões Iniciais da Fase de Planejamento

Questões Iniciais	Respostas
Qual será o objetivo principal da avaliação?	Análise dos problemas que ocorrem no processo de engenharia de requisitos de sistemas embarcados.
Quem será o público alvo da avaliação?	MPEs brasileiras que atuam no setor de desenvolvimento de sistemas embarcados.

O primeiro questionamento foi respondido por meio da análise do foco da pesquisa realizada na área de sistemas embarcados, na qual um dos principais desafios é o atendimento a rígidas restrições de tempo, peso, tamanho, mobilidade, segurança, confiabilidade, consumo de energia e memória (Sommerville 1998). Outro grande obstáculo a ser ultrapassado é a adaptação do software ao hardware e sua separação na modelagem de seus requisitos (Nars 2002).

No que diz respeito à segunda questão, as MPEs foram selecionadas como público alvo da avaliação devido sua efetiva participação no crescimento do mercado brasileiro de desenvolvimento de software e a falta de formalização de seus processos. Segundo pesquisas realizadas pelo Ministério da Ciência e Tecnologia em 2001 (MCT 2002), as MPEs destacam-se no cenário nacional com a participação de 60% da mão de obra efetiva do setor de desenvolvimento de software, onde aproximadamente 70% conhecem algum modelo de qualidade (ISO/IEC122007, ISO 9000 ou CMMI), mas apenas 20% delas utilizam algum desses modelos.

Após a definição do objetivo e do público alvo, o próximo passo foi a seleção das equipes participantes do programa de medições. A Tabela 8 apresenta as equipes avaliadora e avaliadas:

Tabela 8 - Equipe GQM

Equipe de Definição e Coordenação	
Papel	Área de Atuação
Orientadora	Definição de Processos
Pesquisadora	Requisitos
Pesquisadora	Requisitos
Equipes Avaliada	
Papel	Área de Atuação
Respondente	Desenvolvimento de Sistemas de Segurança
Respondente	Desenvolvimento de Sistemas Agrários

Os participantes das equipes avaliadas são funcionários de empresas desenvolvedoras de sistemas embarcados para os mais diversos fins, como desenvolvimento de sistemas agrários e sistemas de segurança. A diversidade dos objetivos do produto final foi propositalmente considerada, pois desta forma a avaliação poderá comparar os resultados e até mesmo as dificuldades encontradas em cada um dos segmentos das organizações participantes do programa. Desta forma, a avaliação torna-se mais isenta e imparcial, sem desvios causados pelo domínio de negócio de uma determinada empresa.

Levando em consideração a distribuição geográfica das MPEs avaliadas, foi acordado que a iteração entre as equipes avaliadora e avaliada seria realizada por e-mail. Devido a essa característica, foi necessária a elaboração da avaliação em um formato que proporcionasse ao usuário facilidade na seleção das respostas de cada questão. Além disso, uma seção de ajuda foi elaborada com o objetivo de melhorar a compreensão do questionário.

Visto que a pesquisa proposta é um programa de medições que não possui o objetivo de melhoria de processo de determinada empresa, não foram avaliados projetos específicos e nenhum dos Planos do modelo GQM foi elaborado. Diante desse contexto, a próxima atividade realizada no programa de medições é a definição do objetivos de negócio e de seus sub-objetivos.

6.1.2 Definição

Conforme exposto no capítulo 5, a definição dos objetivos e sub-objetivos do programa de avaliação deve ser executado etapas. Nessa pesquisa dividimos a fase de definição em três macro-atividades:

- identificar os objetivos da avaliação,
- identificar sub-objetivos;
- identificar entidades e atributos relacionados aos sub-objetivos (questões e métricas).

Na primeiro passo, os objetivos da medição devem ser formalmente definidos e estruturados. Conforme descrito no preâmbulo da seção, o objetivo da avaliação foi um ponto extremante discutido no início do programa. Logo, nessa fase da avaliação a formalização deste objetivo foi realizada. A Tabela 9 foi utilizada como ferramenta de apoio na formalização do objetivo da pesquisa.

Tabela 9 – Modelo de Definição dos Objetivos da Medição (Laboratory s.d.)

Diretrizes	Descrições	Respostas
Analisar:	o objeto sob medição	processo de engenharia de requisitos
Com o propósito de:	entender, controlar e melhorar o objeto de medição.	adequar
Com respeito a:	foco da qualidade do objeto de medição.	aplicabilidade
Do ponto de vista de:	as pessoas que medem o objeto	Equipe de desenvolvimento
No contexto:	o ambiente que as medições acontecerão.	Das MPEs desenvolvedoras de sistemas embarcados.

A Tabela 10 descreve o objetivo do negócio definido com base nas características identificadas na tabela acima.

Tabela 10 - Objetivo do Negócio

Adequar a aplicabilidade do processo de gerenciamento de requisitos de MPEs desenvolvedoras de sistemas embarcados sob o ponto de vista da equipe de desenvolvimento.

Uma vez definido o objetivo do programa de medições, o próximo passo é a identificação detalhada do que se quer saber ou aprender (sub-objetivos). Ao responder o que é necessário e o que se quer saber, os pontos influenciados ou gerenciados pelo objetivo de avaliação serão identificados.

Para facilitar o entendimento e a elaboração da avaliação os sub-objetivos identificados no programa de medições não seguiram o padrão estipulado pelo método GQM descrito no capítulo 5. Ou seja, os sub-objetivos foram convertidos para pontos de verificação (classificações) que classificam as questões e métricas elaboradas na pesquisa.

Os pontos identificados no programa de medições são relacionados as principais características e dificuldades relatadas na literatura sobre o processo de engenharia de requisitos de sistemas embarcados e até mesmo dos sistemas tradicionais, conforme relatado nos capítulos 2 e 3. Logo, eles possuem seu foco em todas as atividades que são executadas durante o processo de engenharia de requisitos considerando que os requisitos funcionais e não-funcionais devem ser levantados e analisados nesse processo.

Outros aspectos analisados na identificação dos pontos de verificação foram os resultados esperados do processo de Gerência de Requisitos do nível G do programa de Melhoria de Processo do Software Brasileiro – MPS.BR. Eles estabelecem os resultados a serem obtidos com a efetiva implementação do processo de Gerência de Requisitos.

A Tabela 11 apresenta todos os pontos de verificação identificados no programa de medições.

Tabela 11 - Pontos de Verificação

Pontos de Verificação
1. Contextualização da empresa.
2. Contextualização do software.
3. Contextualização da equipe de requisitos.
4. O entendimento dos requisitos (funcionais e não-funcionais) é obtido junto aos fornecedores de requisitos?
5. As necessidades, expectativas e restrições do cliente, tanto do produto quanto de suas interfaces, são identificadas e registradas?
6. Os requisitos de software são aprovados?
7. Um conjunto definido de requisitos do cliente é especificado a partir das necessidades, expectativas e restrições identificadas?
8. Um conjunto de requisitos funcionais e não-funcionais do produto e dos componentes do produto que descrevem a solução do problema a ser resolvido, é definido e mantido a partir dos requisitos do cliente?
9. É utilizado algum método, modelo ou linguagem para especificação dos requisitos (funcionais e não-funcionais)?
10. É estabelecida a rastreabilidade bidirecional entre os requisitos e os produtos de trabalho? Em caso positivo, esta rastreabilidade é mantida?

11. Interfaces internas e externas do produto e de cada componente do produto são definidas?
12. Os requisitos são analisados para assegurar que sejam necessários, corretos, testáveis e suficientes para balancear as necessidades dos interessados com as restrições existentes?
13. É considerada e avaliada a dependência entre os requisitos?
14. É considerada e validada a viabilidade dos itens de software atenderem seus requisitos alocados?
15. São feitas revisões nos planos e produtos de trabalho do projeto visando identificar e corrigir inconsistências em relação aos requisitos?
16. As mudanças nos requisitos são gerenciadas ao longo do tempo?

Cada um dos pontos descritos na Tabela 11 foram detalhados na etapa de planejamento, por meio da elaboração de questões que provêm a correta interpretação dos pontos de verificação e dos objetivos relacionados.

6.1.3 Elaboração das Questões e Métricas

A próxima etapa do modelo GQM é a definição das questões e métricas relacionadas aos pontos de verificação. Seguindo as definições do capítulo 5, as questões foram elaboradas em um nível intermediário de abstração, pois elas servirão como guia para a elaboração de suas métricas. Para isso, cada questão foi definida com opções de respostas numeradas que permitem a interpretação quantitativa de suas respectivas métricas. As seções seguintes apresentarão as questões e métricas definidas neste programa de medições.

6.1.3.1 Ponto de Verificação 1

A contextualização da empresa é um aspecto de vital importância para a avaliação, visto que é necessário conhecer o ambiente no qual as equipes avaliadas estão inseridas. Para isso, foram definidas três questões cujas respostas são descritivas e não possuem métricas correlacionadas, portanto, não estão de acordo com as definições do modelo GQM e não geram respostas quantitativas.

De acordo com a Tabela 12, o objetivo das questões definidas para o primeiro ponto de verificação é conhecer o ramo de atuação das empresas avaliadas, seu tempo de experiência no mercado e sua quantidade de funcionários. Essa informações foram contempladas a fim de garantir que todos os participantes da avaliação possuem um de seus segmentos de atuação na área de desenvolvimento de sistemas embarcados. Além disso, o ramo de atuação e o tempo de experiência das empresas auxiliam também na

identificação da criticidade dos sistemas desenvolvidos, o que pode impactar na necessidade de um maior enfoque na definição dos requisitos funcionais e não-funcionais dentro do processo de engenharia de requisitos. Por fim, a quantidade de funcionários da empresa visa identificar seu enquadramento como MPE.

Tabela 12 - Contextualização da Empresa

Ponto de Verificação (PVn)	Questão (QTn)
PV01. Contextualização - A empresa	QT01. Qual o domínio de negócio da empresa?
	QT02. Há quanto tempo a empresa atua neste domínio de negócio?
	QT03. Qual o número de funcionários da empresa?

6.1.3.2 Ponto de Verificação 2

Este ponto de verificação visa esclarecer o contexto do software desenvolvido pelas empresas que participam da avaliação. Suas questões identificam de onde vêm os requisitos do software (cliente interno ou externo) e características de portabilidade e integração. Os indicadores gerados a partir destas questões permitirão reflexões acerca das restrições às quais o software pode estar submetido como plataforma, interface com outros sistemas, padrões e linguagens de desenvolvimento. A Tabela 13 descreve as questões, suas opções de resposta e as métricas associadas à contextualização do software.

Tabela 13 - Contextualização do Software

Ponto de Verificação (PVn)	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão (Mn)	Métrica do Ponto de Verificação (MTn)
PV02. Contextualização – O software	QT04. O software embarcado desenvolvido pela empresa é...	(0) Componente de um produto específico da própria empresa (cliente interno).	Índice de caracterização do software Fórmula: M4 = x, onde x ∈ A / A = {0, 1, 2, 3}	Índice de contextualização do software Fórmula ¹ : MT2 = M4 + M5 + M6
		(1) Componente de um produto específico de outra empresa (cliente externo).		
		(2) Um produto da empresa (cliente interno).		
		(3) Um produto da empresa desenvolvido sob encomenda (cliente externo).		
	QT05. O software embarcado desenvolvido pela empresa possui integração com outros sistemas?	(0) Não possui integração com outros sistemas.	Nível de integração exigido pelo software Fórmula: M5 = x, onde x ∈ A / A = {0, 1, 2, 3}	
		(1) Possui integração com outros sistemas da própria empresa.		
		(2) Possui integração apenas com sistemas de outros fabricantes.		
	QT06. O software embarcado desenvolvido pela empresa deve executar em	(0) O software é dedicado a uma plataforma específica da empresa.	Nível de portabilidade exigido pelo software	
		(0) O software é dedicado a uma plataforma específica de outro fabricante.		

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

	executar em diferentes plataformas?	(1) O software executa em diferentes plataformas da própria empresa.	Fórmula: $M6 = x$, onde $x \in A / A = \{0, 1, 2, 3, 4\}$	
		(2) O software executa em diferentes plataformas de outro fabricante.		
		(3) O software executa em diferentes plataformas da própria empresa e de outro fabricante.		
		(4) O software executa em plataformas de diferentes fabricantes.		

6.1.3.3 Ponto de Verificação 3

A contextualização da equipe de requisitos avalia a existência, tamanho, conhecimento, nível de dedicação e formação acadêmica da equipe responsável pelo levantamento de requisitos. O entendimento destas características auxiliará na modelagem do processo de engenharia de requisitos, de maneira que se possa definir o nível de detalhamento, a quantidade e a designação dos papéis, artefatos e atividades adequados ao desenvolvimento de softwares embarcados em MPEs.

A Tabela 14 descreve as questões, as opções de resposta e as métricas associadas ao ponto de verificação descrito nesta seção.

Tabela 14 - Contextualização da Equipe de Requisitos

Ponto de Verificação (PVn)	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão (Mn)	Métrica do Ponto de Verificação (MTn)
PV03. Contextualização – A Equipe de Requisitos	QT07. No caso do software ser um componente de um produto da própria empresa ou de outro fabricante, existe uma equipe separada para o levantamento dos	(0) O software não é um componente do produto. Ele é o próprio produto da empresa.	Nível de especialização da equipe e do processo de levantamento de requisitos. Fórmula: $M7 = x$, onde $x \in A / A = \{0, 1, 2, 3, 4\}$	Índice de contextualização da equipe de requisitos. Fórmula ¹ : $MT3 = M7 + M8 + M9 + M10 + M11$
		(1) Não existe separação entre as equipes do produto e do software. O levantamento é realizado em conjunto, no mesmo processo.		
		(2) Não existe separação entre as equipes. Mas os levantamentos do produto e do software são realizados em momentos distintos e por processos distintos.		

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

	requisitos do software?	(3) Existe uma equipe separada, responsável pelo levantamento dos requisitos do software. O processo de levantamento é o mesmo do produto.	
		(4) Existe uma equipe separada para o levantamento dos requisitos do software. O processo de levantamento é distinto daquele utilizado para o produto.	
QT08. Caso exista uma equipe separada para os requisitos do software, qual o seu tamanho, em relação ao tamanho total da equipe responsável pelo projeto?	(0) Não existe uma equipe separada.	Tamanho da equipe de requisitos em relação à equipe de projeto. Fórmula: $M8 = x$, onde $x \in A / A = \{0, 1, 2, 3, 4\}$	
	(1) Até 20% da equipe de projeto.		
	(2) De 20 a 30% da equipe de projeto.		
	(3) De 30 a 40% da equipe de projeto.		
QT09. Caso exista uma equipe separada para os requisitos do software, seus membros são dedicados a esta atividade?	(0) Não existe uma equipe separada.	Nível de dedicação da equipe de requisitos. Fórmula: $M9 = x$, onde $x \in A / A = \{0, 1, 2\}$	
	(1) Os membros da equipe de requisitos também desempenham outros papéis ao longo do projeto.		
	(2) Os membros da equipe de requisitos são dedicados a esta atividade.		
QT10. Qual a formação acadêmica dos membros da equipe de requisitos (mesmo que não exista uma equipe separada para tal)?	(0) Os membros da equipe não possuem formação em Ciência da Computação (ou áreas afins) e nem conhecimento da Engenharia de Requisitos.	Nível de formação da equipe de requisitos (Nível de conhecimento da Engenharia de requisitos e Área de formação da equipe).	
	(1) Os membros da equipe não possuem formação em Ciência da Computação (ou áreas afins), mas possuem conhecimento da Engenharia de Requisitos.		

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

		(2) Os membros da equipe possuem formação em Ciência da Computação (ou áreas afins), mas não têm conhecimento da Engenharia de Requisitos.	Fórmula: $M10 = x$, onde $x \in A$ / $A = \{0, 1, 2, 3, 4\}$	
		(3) Os membros da equipe possuem formação em Ciência da Computação (ou áreas afins), e possuem conhecimento superficial da Engenharia de Requisitos.		
		(4) Os membros da equipe possuem formação em Ciência da Computação (ou áreas afins), e possuem conhecimentos profundos da Engenharia de Requisitos.		
	QT11. Qual o nível de conhecimento da equipe de requisitos no que diz respeito ao negócio do software a ser desenvolvido?	(0) A equipe de requisitos não possui conhecimento do negócio.	Nível de conhecimento da equipe no negócio do software a ser desenvolvido. Fórmula: $M11 = x$, onde $x \in A$ / $A = \{0, 1, 2\}$	
		(1) A equipe de requisitos tem conhecimento superficial do negócio.		
		(2) A equipe de requisitos tem conhecimentos aprofundados do negócio.		

6.1.3.4 Pontos de Verificação 4 e 9

O modo como os requisitos são levantados, os níveis de detalhe e de especialização do responsável pelo levantamento de requisitos são aspectos que avaliam se o entendimento dos requisitos funcionais e não-funcionais é obtido juntos aos fornecedores de requisitos, conforme descrito na Tabela 15. A resposta dessas questões gera indicadores que auxiliam na definição dos papéis, do nível de detalhamento dos artefatos e das técnicas de levantamento de requisitos que melhor se adaptem as características dos tipos de empresa e sistemas definidos nesse estudo.

Tabela 15 - Entendimento dos Requisitos Funcionais e Não-Funcionais

Ponto de Verificação (PVn)	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão (Mn)	Métrica do Ponto de Verificação (MTn)
PV04. O entendimento	QT12. Nível de	(0) Os requisitos não são levantados	Nível de entendimento	Nível de entendimento

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

dos requisitos (funcionais e não-funcionais) é obtido junto aos fornecedores de requisitos?	entendimento dos requisitos.	(1) Os requisitos são levantados informalmente (conversa informal).	dos requisitos.	dos requisitos funcionais e não-funcionais	
		(2) Os requisitos são recebidos já levantados.			Fórmula: $M12 = x$, onde $x \in A / A = \{0, 1, 2, 3\}$
		(3) Os requisitos são levantados por meio de reuniões formais.			Fórmula ¹ : $MT4 = M12 + M13 + M14$
	QT13. Nível de especialização da equipe de levantamento de requisitos nesta atividade.	(0) Os requisitos não são levantados.	Nível de especialização da equipe de levantamento de requisitos nesta atividade.		
		(1) Os requisitos são levantados pela equipe de implementação.			
		(1) Os requisitos são levantados pelo gerente de projetos.			
		(2) Os requisitos são levantados pelos arquitetos ou projetistas.			
		(3) Os requisitos são levantados pela equipe de requisitos.			
	QT14. Nível de detalhe do levantamento de requisitos.	(0) As necessidades do cliente não são identificadas.	Nível de detalhe do levantamento de requisitos.		
		(1) As necessidades do cliente são identificadas.			
		(2) As necessidades do cliente e as restrições do sistema são identificadas.			
		(3) As necessidades do cliente, as restrições do sistema e as integrações com outros sistemas/dispositivos são identificadas.			
PV09. É utilizado algum método, modelo ou linguagem para especificação dos requisitos (funcionais e não-funcionais)?	QT20. Qual o nível de utilização de métodos/módulos no levantamento/especificação dos requisitos?	(0) Os requisitos funcionais e não-funcionais não são especificados.	Nível de utilização de métodos/módulos no levantamento/especificação dos requisitos (funcionais e não-funcionais).	Índice de utilização de ferramentas de apoio. Fórmula ¹ :	
		(1) Os requisitos funcionais e não-funcionais são levantados/especificados sem utilização de técnicas auxiliares.			

¹:Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

não-funcionais)?	requisitos (funcionais e não-funcionais)?	(2) Os requisitos funcionais e não-funcionais são levantados/especificados formalmente com o auxílio de técnicas auxiliares (<i>checklist</i> , <i>brainstorm</i> e etc.).	(funcionais e não-funcionais). Fórmula: $M20 = x$, onde $x \in A / A = \{0, 1, 2\}$	MT9 = M20
------------------	-------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------	-----------

6.1.3.5 Pontos de Verificação 5 e 6

A compreensão de como é realizada a aprovação dos requisitos, a identificação e o registro das necessidades, expectativas e restrições do cliente é necessária para a correta definição dos artefatos a serem gerados no processo de engenharia de requisitos e dos elementos que o compõem. Além disso, o registro e aprovação apropriada das informações levantadas assegura que nenhum requisito seja ignorado ou entendido de forma inadequada. A Tabela 16 apresenta as questões e métricas relacionadas a este ponto de verificação.

Tabela 16 – Identificação e Registro das Necessidades, Expectativas e Restrições do Cliente

Ponto de Verificação (PVn)	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão (Mn)	Métrica do Ponto de Verificação (MTn)
PV05. As necessidades, expectativas e restrições do cliente, tanto do produto quanto de suas interfaces, são identificadas e registradas?	QT15. Qual o nível de formalização dos requisitos?	(0) Os requisitos não são levantados.	Nível de formalização dos requisitos. Fórmula: $M15 = x$, onde $x \in A / A = \{0, 1, 2, 3, 4\}$	Nível de identificação e registro das necessidades, expectativas e restrições do cliente. Fórmula ¹ : $MT5 = M15 + M16$
		(1) Os requisitos são acordados informalmente (verbalmente, telefone) mas não são registrados.		
(2) Os requisitos são acordados e registrados informalmente (ex: e-mail, documento desestruturado).				
(3) Os requisitos são acordados e registrados em documentos estruturados (ex: atas de reunião).				
		(4) Os requisitos são acordados e registrados formalmente (ex: documento de especificação de requisitos).		
	QT16. Qual o nível de formalização	(0) As necessidades, expectativas e restrições do cliente não são levantadas.	Nível de formalização do	

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

	do detalhamento do levantamento /especificação de requisitos?	<p>(1) As necessidades, expectativas e restrições do cliente são levantadas informalmente e não são registradas.</p> <p>(2) As necessidades, expectativas e restrições do cliente são levantadas e registradas em documentos informais (e-mail).</p> <p>(3) As necessidades, expectativas e restrições do cliente são levantadas e registradas em atas de reunião.</p> <p>(4) As necessidades, expectativas e restrições do cliente são levantadas e registradas em documentos formais do processo (Ex. Documento de Visão).</p>	<p>detalhamento da levantamento /especificação de requisitos.</p> <p>Fórmula: $M16 = x$, onde $x \in A$ / $A = \{0, 1, 2, 3, 4\}$</p>	
PV06. Os requisitos de software são aprovados?	QT17. Qual o nível de formalização dos requisitos?	<p>(0) Os requisitos não são levantados.</p> <p>(1) Os requisitos são acordados informalmente (verbalmente, telefone) mas não são registrados.</p> <p>(2) Os requisitos são acordados e registrados informalmente (ex: e-mail, documento desestruturado).</p> <p>(3) Os requisitos são acordados e registrados em documentos estruturados (ex: atas de reunião).</p> <p>(4) Os requisitos são acordados e registrados formalmente (ex: documento de especificação de requisitos).</p>	<p>Nível de aprovação dos requisitos.</p> <p>Fórmula: $M17 = x$, onde $x \in A$ / $A = \{0, 1, 2, 3, 4\}$</p>	<p>Nível de aprovação dos requisitos.</p> <p>Fórmula¹: $MT6 = M17$</p>

6.1.3.6 Ponto de Verificação 7

Outro assunto que foi ressaltado na avaliação é o modo como os requisitos são especificados, uma vez que uma das principais causas de falha do desenvolvimento de software é o levantamento deficiente de seus requisitos (Hofmann 2001). Conseqüentemente, é necessário que os mesmos sejam identificados a partir das necessidades do cliente, que são refinadas em características e posteriormente em

¹:Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

requisitos do sistema. A Tabela 17 apresenta as questões e métricas definidas para este ponto de verificação.

Tabela 17 - Modo de Especificação dos Requisitos

Ponto de Verificação (PVn)	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão (Mn)	Métrica do Ponto de Verificação (MTn)
PV07. Um conjunto definido de requisitos do cliente é especificado a partir das necessidades, expectativas e restrições identificadas?	QT18. Qual o nível de refinamento dos requisitos funcionais?	(0) Os requisitos são diretamente levantados/definidos.	Nível de refinamento dos requisitos funcionais. Fórmula: $M18 = x$, onde $x \in A$ / $A = \{0, 1, 2, 3\}$	Nível de refinamento dos requisitos funcionais. Fórmula ¹ : $MT7 = M18$
		(1) As necessidades são refinadas somente em nível de características.		
		(2) As necessidades são refinadas diretamente em nível de requisitos.		
		(2) As características são diretamente levantadas e refinadas em nível de requisitos.		
		(3) As necessidades são refinadas em nível de características e posteriormente de requisitos.		

6.1.3.7 Pontos de Verificação 8 e 14

Para garantir que o sistema atenda a todos os seus requisitos é necessário que haja um mapeamento (ligação, dependência) entre os requisitos funcionais e não funcionais, de maneira que não sejam definidos requisitos inviáveis para a estrutura física do sistema. Além disso, as restrições referentes aos requisitos não-funcionais e sua relação com os requisitos funcionais são características marcantes no desenvolvimento de sistemas embarcados. Dessa forma a Tabela 18 define questões, suas opções de resposta e métricas que avaliam o relacionamento entre os requisitos funcionais e não-funcionais tanto na sua identificação como manutenção, bem como, a sua viabilidade de implementá-los.

Tabela 18 – Definição e Manutenção dos Requisitos Funcionais e Não – Funcionais

Ponto de Verificação (PVn)	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão (Mn)	Métrica do Ponto de Verificação (MTn)
PV08. Um conjunto de requisitos	QT19. Qual o nível de entendimento	(0) Os requisitos não-funcionais não são identificados.	Nível de definição e manutenção	Nível de definição e manutenção dos

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

funcionais e não-funcionais do produto e dos componentes do produto que descrevem a solução do problema a ser resolvido, é definido e mantido a partir dos requisitos do cliente?	dos requisitos não-funcionais?	(1) Os requisitos não-funcionais são identificados, mas não são definidos e mantidos a partir dos requisitos funcionais identificados.	dos requisitos funcionais e não-funcionais. Fórmula: M19 = x, onde x ∈ A / A = {0, 1, 2, 3, 4}	requisitos funcionais e não-funcionais. Fórmula ¹ : MT8 = M19
		(2) Os requisitos não-funcionais são identificados e são definidos a partir dos requisitos funcionais identificados.		
		(3) Os requisitos não-funcionais são identificados e são definidos e mantidos a partir dos requisitos funcionais identificados.		
		(4) Os requisitos não-funcionais são identificados e são definidos e mantidos a partir dos requisitos funcionais identificados e “comparados” aos próprios requisitos funcionais levantados.		
PV14. É considerada e validada a viabilidade dos itens de software atenderem seus requisitos alocados?	QT26. Qual o nível de análise da viabilidade dos requisitos?	(0) A viabilidade dos requisitos funcionais e não-funcionais não é avaliada.	Nível de análise da viabilidade dos requisitos. Fórmula: M26 = x, onde x ∈ A / A = {0, 1, 2}	Nível de análise da viabilidade dos requisitos. Fórmula ¹ : MT14 = M26
		(1) A viabilidade dos requisitos funcionais é avaliada.		
		(1) A viabilidade dos requisitos não-funcionais é avaliada.		
		(2) A viabilidade dos requisitos funcionais e não-funcionais é avaliada.		

6.1.3.8 Pontos de Verificação 10 e 13

A rastreabilidade é um mecanismo que facilita a avaliação dos impactos de mudanças nos sistemas, por meio do mapeamento da dependência entre seus requisitos, planos de projeto e produtos de trabalho. A rastreabilidade horizontal estabelece a dependência entre os requisitos e a rastreabilidade vertical permite o mapeamento das necessidades, características e requisitos levantados até o nível de decomposição mais baixo do produto (Softex s.d.). Dessa forma, os pontos de verificação descritos na Tabela 19 visam determinar como as empresas participantes da avaliação tratam a rastreabilidade e a dependência de seus requisitos.

Tabela 19 - Rastreabilidade dos Requisitos

Ponto de	Questão	Opções de Resposta (A)	Métrica da	Métrica do
----------	---------	------------------------	------------	------------

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

Verificação (PVn)	(QTn)		Questão (Mn)	Ponto de Verificação (MTn)
PV10. É estabelecida a rastreabilidade bidirecional entre os requisitos e os produtos de trabalho? Em caso positivo, esta rastreabilidade é mantida?	QT21. Qual o nível de rastreabilidade e dos requisitos?	(0) A rastreabilidade entre os requisitos e os produtos de trabalhos não é elaborada.	Nível de rastreabilidade e dos requisitos. Fórmula: $M21 = x$, onde $x \in A / A = \{0, 1, 2\}$	Nível de rastreabilidade dos requisitos. Fórmula ¹ : $MT10 = M21$
		(1) A rastreabilidade entre os requisitos de trabalho é elaborada.		
		(2) A rastreabilidade entre os requisitos e os produtos de trabalhos é elaborada e mantida conforme o andamento do projeto.		
PV13. É considerada e avaliada a dependência entre os requisitos?	QT25. Qual o nível de análise da dependência entre os requisitos?	(0) A dependência entre os requisitos não é avaliada.	Nível de análise da dependência entre os requisitos. Fórmula: $M25 = x$, onde $x \in A / A = \{0, 1, 2\}$	Nível de análise da dependência entre os requisitos. Fórmula ¹ : $MT13 = M25$
		(1) A dependência entre os requisitos funcionais é avaliada.		
		(1) A dependência entre os requisitos não-funcionais é avaliada.		
		(2) A dependência entre os requisitos funcionais e não-funcionais é avaliada.		

6.1.3.9 Ponto de Verificação 12

As questões e métricas relacionadas este ponto de verificação são descritas na Tabela 20, elas têm como objetivo analisar quais são os papéis envolvidos na realização dos testes e como são executadas as validações dos requisitos. Dessa forma será possível adaptar as estratégias de teste ao contexto das empresas avaliadas, sem perder o foco da integridade do sistema.

Tabela 20 – Análise dos Requisitos

Ponto de Verificação (PVn)	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão (Mn)	Métrica do Ponto de Verificação (MTn)
PV12. Os requisitos são analisados para assegurar que sejam necessários, corretos, testáveis e suficientes para	QT23. Qual o nível de análise de erros nos requisitos?	(0) Os requisitos não são testados pela equipe de desenvolvimento nem validados pelo cliente?	Nível de análise de erros nos requisitos. Fórmula:	Índice de corretude dos requisitos. Fórmula ¹ : $MT12 = M23 + M24$
		(1) Os requisitos são testados pela equipe de desenvolvimento e não são validados pelo cliente.		

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

suficientes para balancear as necessidades dos interessados com as restrições existentes?		(1) Os requisitos são validados pelo cliente e não são testados pela equipe de desenvolvimento.	M23 = x, onde $x \in A / A = \{0, 1, 2\}$	M24
		(2) Os requisitos são testados pela equipe de desenvolvimento e validados pelo cliente.		
	QT24. Qual o nível de corretude dos testes de requisitos?	(0) Os requisitos não são testados.	Nível de corretude dos testes de requisitos. Fórmula: M24 = x, onde $x \in A / A = \{0, 1, 2, 3\}$	
		(1) Os requisitos são testados unitariamente e/ou integralmente pela equipe de desenvolvimento sem a utilização de casos de testes.		
		(2) Os requisitos são testados unitariamente pela equipe de desenvolvimento com a utilização de casos de testes.		
		(2) Os requisitos são testados integralmente pela equipe de desenvolvimento com a utilização de casos de testes.		
		(3) Os requisitos são testados unitariamente e integralmente pela equipe de desenvolvimento com a utilização de casos de testes.		

6.1.3.10 Pontos de Verificação 15 e 16

Outro ponto que merece atenção é o acompanhamento das mudanças nos requisitos do projeto. Alterações nos requisitos estão entre os pontos que mais acarretam falhas nos projetos de desenvolvimento de sistemas embarcados (Carro 2002), portanto, eles devem ser gerenciados ao longo do processo. Neste sentido, é fundamental garantir que os planos e produtos de trabalho do projeto estejam sendo revisados, a fim de que reflitam os requisitos definidos. Estes artefatos serão importantes para a tomada de decisões em relação ao projeto e para futuras manutenções no sistema. Tais aspectos são facilmente atendidos quando se realiza a rastreabilidade dos requisitos e produtos de trabalho. Sendo assim, os indicadores gerados a partir das métricas apresentadas na Tabela 21 permitem analisar estes aspectos nas empresas avaliadas.

Tabela 21 – Revisão dos Planos e Produtos de Trabalho

Ponto de Verificação	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão	Métrica do Ponto de
----------------------	---------------	------------------------	--------------------	---------------------

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

(PVn)			(Mn)	Verificação (MTn)
PV15. São feitas revisões nos planos e produtos de trabalho do projeto visando identificar e corrigir inconsistências em relação aos requisitos?	QT27. Qual o nível de análise de revisões e correções dos produtos de trabalho?	(0) Não são realizadas revisões dos planos e produtos de trabalho.	Nível de análise de revisões e correções dos produtos de trabalho. Fórmula: $M27 = x$, onde $x \in A / A = \{0, 1, 2\}$	Nível de análise de revisões e correções dos produtos de trabalho. Fórmula ¹ : $MT15 = M27$
		(1) São realizadas revisões e correções dos planos e produtos de trabalho durante o processo de desenvolvimento do sistema.		
		(1) São realizadas revisões e correções dos planos e produtos de trabalho durante o processo de manutenção do sistema.		
		(2) São realizadas revisões e correções dos planos e produtos de trabalho durante o processo de desenvolvimento e manutenção do sistema.		
PV16. As mudanças nos requisitos são gerenciadas ao longo do tempo?	QT28. Qual o nível de análise de manutenções nos requisitos funcionais e não-funcionais?	(0) Os requisitos funcionais e não-funcionais não são mantidos.	Nível de análise de manutenções nos requisitos funcionais e não-funcionais. Fórmula: $M28 = x$, onde $x \in A / A = \{0, 1, 2\}$	Nível de análise de manutenções nos requisitos funcionais e não-funcionais. Fórmula: $MT16 = M28$
		(1) Os requisitos funcionais são mantidos ao longo do tempo		
		(1) Os requisitos não-funcionais são mantidos ao longo do tempo.		
		(2) Os requisitos funcionais e não-funcionais são mantidos ao longo do tempo.		

6.1.3.11 Ponto de Verificação 11

A identificação e o registro dos usuários e dos sistemas que interagem com o software em desenvolvimento auxilia na definição da sua arquitetura e de seus protocolos de comunicação, além de facilitar a obtenção de requisitos ainda não levantados. A Tabela 22 apresenta as questões, opções de respostas e métricas relacionadas a esse ponto de verificação.

Tabela 22 - Identificação das Interfaces do Sistema

Ponto de Verificação (PVn)	Questão (QTn)	Opções de Resposta (A)	Métrica da Questão (Mn)	Métrica do Ponto de Verificação (MTn)
PV11. Interfaces internas e	QT22. Qual o nível de	(0) Os atores do sistema não são definidos.	Nível de identificação	Nível de identificação de

¹: Interpretação - Quanto maior o valor de MTn, melhor é o índice/nível.

externas do produto e de cada componente do produto são definidas?	identificação de interfaces?	(1) Os atores do sistema são definidos, mas não são registrados.	de interfaces.	interfaces.	
		(2) Os atores do sistema são definidos e registrados em documentos informais (Ex.: e-mail).			Fórmula: $M22 = x$, onde $x \in A / A = \{0, 1, 2, 3, 4\}$
		(3) Os atores do sistema são definidos e registrados em documentos estruturados (Ex. Ata de reunião).			Fórmula ¹ : $MT11 = M22$
		(4) Os atores do sistema são definidos e registrados em documentos formais (ex: documento de visão).			

Outros aspectos foram avaliados para este objetivo de medição, como, o modo de elaboração dos cenários operacionais e a análise de viabilidade de operação e manutenção dos softwares embarcados. No entanto, ao realizar a interpretação dessas questões, considerou-se que esses pontos não fazem parte do processo de engenharia de requisitos, e conseqüentemente não são relacionados ao objetivo da medição.

A Figura 19 apresenta a relação entre o objetivo de medição, os itens de verificação, e suas questões.

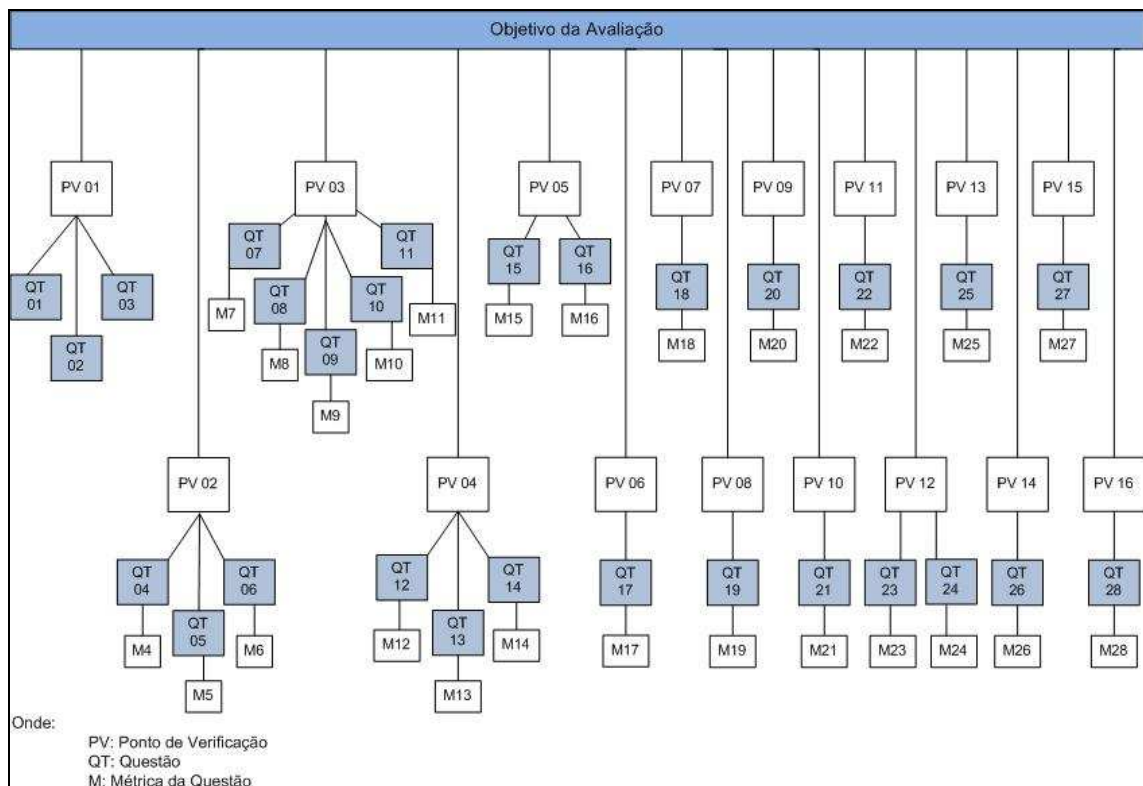


Figura 19 - Relação entre os Elementos do Modelo GQM

Por fim foi realizada a revisão do questionário que é a última atividade prevista na fase de definição do modelo GQM. Após finalizar sua elaboração, os pesquisadores e a orientadora efetuaram uma revisão formal que apontou melhorias no questionário. Além disso, a avaliação foi submetida a análise de todos os participantes, que puderam sugerir novas modificações.

6.1.4 Coleta

Conforme descrito no capítulo 5, a fase de coleta do modelo GQM é caracterizada pela obtenção das respostas do questionário de avaliação. O procedimento de coleta de dados foi definido considerando o contexto das equipes respondentes, com o objetivo de maximizar a eficiência e eficácia da avaliação. Sendo assim, a obtenção das respostas foi realizada manualmente, por meio do preenchimento de planilhas Excel.

A primeira atividade dessa fase é a validação dos procedimentos e das ferramentas de coleta. Dada a distância entre os participantes do programa de medições não foi realizada uma reunião formal para sua validação. No entanto, como descrito na seção anterior, o questionário foi submetido a toda equipe para sugestões de melhorias que foram incorporadas.

Após as correções para aprimoramento do questionário, assumiu-se que o mesmo atendia aos objetivos do programa de medições. Por conseguinte, foi iniciada a segunda atividade desta fase que consiste na distribuição do questionário a fim de coletar as respostas para formação da base de métricas.

Durante a fase de coleta ocorreram alguns problemas de comunicação com os participantes da avaliação no envio e recebimento dos questionário. Como o prazo para finalizar o programa de medições era exíguo não foi possível extê-lo, de maneira que a interpretação foi realizada com a resposta de apenas um questionário.

Os dados coletados foram conferidos em relação a sua completude e armazenados para posterior interpretação. A planilha com as respostas do participante da avaliação está disponível no APÊNDICE A.

6.1.5 Interpretação

Conforme definido no capítulo 5, a fase que sucede a coleta dos dados do programa de medições é a interpretação das informações obtidas. Para execução desta etapa o trabalho foi dividido em três atividades: análise das informações coletadas, cálculo dos dados quantitativos e interpretação dos índices calculados.

6.1.5.1 Análise das Informações

Dessa forma, após o recebimento do questionário respondido pelo participante da avaliação foi realizada uma análise para verificar a coerência entre as respostas.

Outro aspecto identificado na análise das respostas foi a inexatidão do questionário. Uma das preocupações no momento de elaborá-lo foi mantê-lo o mais legível possível devido ao fato de profissionais de diversas áreas participarem da pesquisa. Dessa forma, foi excluída a utilização de termos técnicos da área de informática e foi elaborada uma seção de ajuda para cada uma das questões. No entanto, pôde-se perceber na análise que as respostas de determinadas questões poderiam ser interpretadas de uma maneira por profissionais de engenharia de software e de outra por outros tipos de profissionais. Por exemplo:

Na segunda questão do primeiro ponto de verificação especificado na Tabela 16 é verificado o nível de formalização do levantamento/especificação de requisitos. Quando questionados se a especificação de requisitos era registrada em documentos formais, não foi levado em consideração o nível de detalhe dos requisitos neste documento. Desta forma, a partir da resposta informada, não se consegue obter o grau de entendimento desejado sob o ponto de vista da Engenharia de Requisitos. Visto que o documento utilizado pelo participante é formal, mas pode não conter o nível de detalhe exigido para um documento de especificação de requisitos.

Devido à escassez de tempo para finalização deste estudo optou-se em continuar a interpretação dos dados obtidos. Visto que a elaboração do processo de engenharia de requisitos para MPEs desenvolvedoras de softwares embarcados utilizará, além das métricas obtidas no programa de medições, o embasamento teórico encontrado na literatura.

6.1.5.2 Cálculo dos Dados Quantitativos

Em conformidade com as definições do capítulo 5, a interpretação dos dados coletados foi realizada na direção *bottom-up*. Ou seja, a partir dos índices coletados em cada questão foram calculados os dados quantitativos por ponto de verificação. A Tabela 23 apresenta cada um desses índices.

Tabela 23 - Índices dos Pontos de Verificação

Métrica da Questão (Mn)	Índice¹	Métrica do Ponto de Verificação (MTn)	Índice²
M04. Índice de caracterização do software.	M04 = 2	M2. Índice de criticidade do contexto do software.	MT2 = 5
M05. Nível de integração exigido pelo software.	M05 = 3		
M06. Nível de portabilidade exigido pelo software.	M06 = 0		
M07. Nível de especialização da equipe e do processo de levantamento de requisitos.	M07 = 2	MT3. Índice de contextualização da equipe de requisitos.	MT3 = 7
M08. Tamanho da equipe de requisitos em relação à equipe de projeto.	M08 = 0		
M09. Nível de dedicação da equipe de requisitos.	M09 = 0		
M10. Nível de formação da equipe de requisitos (Nível de conhecimento da Engenharia de requisitos e Área de formação da equipe).	M10 = 3		
M11. Nível de conhecimento da equipe no negócio do software a ser desenvolvido.	M11 = 2	MT4. Nível de entendimento dos requisitos funcionais e não-funcionais.	MT4 = 8
M12. Nível de entendimento dos requisitos.	M12 = 2		
M13. Nível de especialização da equipe de levantamento de requisitos nesta atividade.	M13 = 3		
M14. Nível de detalhe do levantamento de requisitos.	M14 = 3	MT9. Índice de utilização de ferramentas de apoio.	MT9 = 1
M20. Nível de utilização de métodos/modelos no levantamento/especificação dos requisitos (funcionais e não-funcionais).	M20 =		
M15. Nível de formalização dos requisitos.	M15 = 3	MT5. Nível de identificação e registro das necessidades,	MT5 = 6

¹: Índice do Ponto de Verificação (Mn).

²: Índice da Questão (MTn).

M16. Nível de formalização do detalhamento da levantamento/especificação de requisitos.	M16 = 3	expectativas e restrições do cliente.	
M17. Nível de aprovação dos requisitos .	M17 = 4	MT6. Nível de aprovação dos requisitos.	MT6 = 4
M18. Nível de refinamento dos requisitos funcionais.	M18 = 2	MT7. Nível de refinamento dos requisitos funcionais.	MT7 = 2
M19. Nível de Definição e Manutenção dos Requisitos Funcionais e Não-Funcionais.	M19 = 2	MT8. Nível de definição e manutenção dos requisitos funcionais e não-funcionais.	MT8 = 2
M26. Nível de análise da viabilidade dos requisitos.	M26 = 1	MT14. Nível de análise da viabilidade dos requisitos.	MT14 = 1
M21. Nível de rastreabilidade dos requisitos.	M21 = 1	MT10. Nível de rastreabilidade dos requisitos.	MT10 = 1
M25. Nível de análise da dependência entre os requisitos.	M25 = 1	MT13. Nível de análise da dependência entre os requisitos.	MT13 = 1
M23. Nível de análise de erros nos requisitos.	M23 = 1	MT12. Índice de corretude dos requisitos.	MT12 = 2
M24. Nível de corretude dos testes de requisitos.	M24 = 1		
M27. Nível de análise de revisões e correções dos produtos de trabalho.	M27 = 1	MT15. Nível de análise de revisões e correções dos produtos de trabalho.	MT15 = 1
M28. Nível de análise de manutenções nos requisitos funcionais e não-funcionais.	M28 = 2	MT16. Nível de análise de manutenções nos requisitos funcionais e não-funcionais.	MT16 = 2
M22. Nível de identificação de interfaces.	M22 = 3	MT11. Nível de identificação de interfaces.	MT11 = 3

6.1.5.3 Interpretação dos Índices Calculados

A empresa participante da avaliação é uma MPE com 6 funcionários que atua no desenvolvimento de hardware criptográfico há 3 anos. Os softwares embarcados desenvolvidos são dedicados a uma plataforma própria e compõem um produto da empresa. No entanto, possuem integrações não somente com sistemas da empresa, mas também com sistemas de outros fabricantes. Como o software desenvolvido é parte de um produto da empresa avaliada, os requisitos do software são levantamentos/criados internamente.

As métricas obtidas no ponto de verificação que avalia a contextualização do software (Tabela 23, item 0) revelaram que seu nível de portabilidade não é alto, no entanto seu índice de integração com outros sistemas, inclusive de outros fabricantes, é crítico. Logo, o contexto no qual o software está inserido é de média criticidade.

Já os índices de contextualização da equipe de requisitos mostraram que não exista na empresa uma equipe específica para o levantamento de requisitos. No entanto, apesar

do software ser apenas um componente de um produto, seu levantamento de requisitos é realizado por um processo a parte. Embora a formação dos responsáveis pelo levantamento seja em computação ou áreas afins, estes não possuem conhecimentos aprofundados sobre a Engenharia de Requisitos. Em compensação, possuem conhecimentos arraigados acerca do negócio para o qual o software está sendo desenvolvido. O índice final do ponto de verificação descrito no item 0 da Tabela 23 posiciona a equipe em um nível intermediário de contextualização.

A empresa apresentou um excelente nível de entendimento dos requisitos funcionais e não funcionais (Tabela 23, item 0). No entanto, o estudo revelou que o levantamento dos requisitos é realizado pelos arquitetos e projetistas do sistema, de maneira que a equipe de desenvolvimento recebe os requisitos prontos para implementação. E este levantamento é realizado sem a utilização de nenhum método, modelo ou linguagem específica (Tabela 23, item 0).

Apesar de terem mostrado um bom índice na identificação e registro das necessidades, expectativas e restrições do cliente (Tabela 23, item 0), a pesquisa expôs que o registro destas informações não é realizado em um documento específico para esse fim. No entanto, essas informações são acordadas e registradas em documentos estruturados, por exemplo, atas de reunião. Já o índice de aprovação dos requisitos propriamente ditos revelou que os mesmos são acordados e registrados formalmente em documentos como o de especificação de requisitos (Tabela 23, item 0).

Com relação ao item 0 da Tabela 23, que analisa o nível de refinamento dos requisitos funcionais, o índice encontrado revelou que as necessidades do cliente não são refinadas gradualmente. Dessa maneira, as necessidades do cliente são levantadas e refinadas no nível de requisitos sem a preocupação de identificar as características do sistema.

No item 0 da Tabela 23 o cálculo dos dados quantitativos revelou que os requisitos não-funcionais são identificados a partir dos requisitos-funcionais, no entanto, não são mantidos ao longo do tempo. Além disso, são realizados somente estudos de viabilidade dos requisitos funcionais (item 0 da Tabela 23).

O índice que mede a dependência entre os requisitos mostrou que esta é avaliada somente entre os requisitos funcionais, desconsiderando os não-funcionais (item 0 da Tabela 23). Já a rastreabilidade dos requisitos apesar de ser elaborada, não é mantida ao longo do projeto (item 0 da Tabela 23).

No que diz respeito à corretude dos requisitos foi identificado que estes são testados pela equipe de desenvolvimento sem a utilização de roteiros de testes mas não são validados pelo cliente (item 0 da Tabela 23).

Já os dados quantitativos referentes ao as revisões dos produtos de trabalho denunciaram que a sua revisão e correção é realizada durante o processo de desenvolvimento do software mas não durante sua manutenção (item 0 da Tabela 23). Por outro lado, as mudanças dos requisitos funcionais e não funcionais são mantidas ao longo do tempo (item 0 da Tabela 23). É importante ressaltar que essa manutenção dos requisitos não-funcionais não leva em consideração os requisitos funcionais relacionados, conforme interpretação do item 0 da Tabela 23. Vale destacar também que as manutenções mais frequentes na empresa consistem na geração de versões especializadas do produto que não substituem os requisitos originais.

Por fim, podemos identificar por meio dos valores calculados que as interfaces do sistema são representadas com a utilização de atores, no entanto, isto é registrado em documentos sem uma estrutura específica para tal, como atas de reuniões (item 0 da Tabela 23).

6.1.6 *Captura de Experiências*

Como o estudo realizado possui apenas cunho científico e foco do trabalho limita-se a propor adequações no processo de engenharia de requisitos do OpenUP, a captura de experiências não foi realizada.

6.2 Considerações Finais

Apesar do estudo não ter sido realizado conforme o planejamento inicial, a avaliação da organização participante comprovou que muitas das dificuldades no desenvolvimento de sistemas embarcados encontradas na literatura realmente ocorrem no cotidiano das MPEs que desenvolvem este tipo de software.

Como verificado na interpretação do programa de medições os requisitos não-funcionais não recebem o tratamento adequado às características dos softwares embarcados. Uma vez que não é realizada a análise de viabilidade nem é identificada a dependência e rastreabilidade destes requisitos. Em consequência disso, quando realizada a manutenção dos requisitos não é possível analisar a coerência entre eles.

Outro fato comprovado é que os profissionais que realizam o levantamento/especificação dos requisitos não possui conhecimento especializado, além de não serem dedicados exclusivamente a estas tarefas. E talvez, por esse motivo não sejam elaborados documentos adequados ao levantamento/especificação de requisitos, não sejam

utilizadas técnicas ou ferramentas de apoio a estas atividades e o refinamento dos requisitos níveis de não seja executado respeitando os níveis de necessidade, característica e requisitos.

Embora não identificado na literatura pesquisada, outros aspectos como a não utilização de roteiros na execução dos testes do software e a falta de revisões dos produtos de trabalho na fase de manutenção foram encontrados durante a avaliação.

Após a execução do programa de medições, todas as deficiências identificadas servirão de base para proposição de adequações no processo de engenharia de requisitos do PU a fim de adaptá-lo ao contexto das MPEs desenvolveras de software embarcado. Esta atividade será descrita em detalhes no capítulo 7.

7. DEFINIÇÃO DO PROCESSO DE ENGENHARIA DE REQUISITOS PARA O DESENVOLVIMENTO DE SISTEMAS EMBARCADOS

Este capítulo apresentará a elaboração do processo de engenharia de requisitos para o desenvolvimento de sistemas embarcados. Para isso foram utilizadas as informações levantadas nos capítulos 02, 03 e 04, os indicadores gerados no programa e medições descrito no capítulo 06 e o contexto das MPEs brasileiras que constroem esse tipo de software.

A organização do capítulo é realizada da seguinte forma: a seção 7.1 apresenta a elaboração do Processo de Engenharia de Requisitos para Sistemas Embarcados detalhada em termo dos elementos do processo (Papéis, Tarefas, Produtos de Trabalho, Diretrizes, Listas de Verificações e Conceitos). Na seção 7.2 é apresentada uma comparação do ProReqSE com outros processos para sistemas embarcados. A seção 7.3 descreve as considerações finais do capítulo.

7.1 Processo de Engenharia de Requisitos para Sistemas Embarcados – ProReqSE

O Processo de Engenharia de Requisitos para Sistemas Embarcados (ProReqSE) foi modelado por meio da ferramenta *Eclipse Process Framework Composer* (EPF) (Foudation 2008). O EPF é uma ferramenta visual de modelagem de processos que utiliza como base o *Software Process Engineering Metamodel* (SPEM) e possibilita a importação de bibliotecas de processos a fim de personalizá-las de acordo com o domínio da aplicação. O ProReqSE utilizou como base a biblioteca do Open UP. O *Basic Unified Process* foi criado pela IBM e rebatizado de Open Unified Process (OpenUP) em março de 2006. O OpenUp é uma simplificação do RUP que se adapta à realidade de pequenas equipes.

O fato de o EPF ser uma ferramenta *freeware* e o OpenUP uma biblioteca aberta foram aspectos fortemente considerados no momento de escolha da ferramenta e do modelo de processo a ser utilizado na elaboração do ProReqSE. A meta era melhorar a qualidade dos processos atuais sem onerar os custos para sua adoção. Afinal, quanto menores as dificuldades para implementação do ProReqSE, maiores serão suas chances de aceitação.

Da mesma maneira que o OpenUP, o ProReqSE é um processo elaborado para aplicação em equipes pequenas e co-localizadas que buscam uma abordagem simplificada para a especificação de seus produtos. Esta abordagem foi mantida devido ao curto prazo normalmente exigido para o desenvolvimento de sistemas embarcados, conforme descrito no capítulo 02, e ao contexto das MPEs. Este tipo de empresa possui equipes bastante

reduzidas, de maneira que um membro, normalmente, assume mais de um papel dentro do processo, o que demanda, portanto, um processo leve, ágil e eficiente.

Outros dois pontos foram considerados na definição das principais características do ProReqSE: a informalidade dos processos de desenvolvimento das micro e pequenas empresas brasileiras e a formação multi - disciplinar das equipes de desenvolvimento de sistemas embarcados. Dessa forma, o ProReqSE é:

- Personalizável: de maneira que as empresas possam adequá-lo às suas necessidades de acordo com sua cultura e nível de formalização;
- Completo: permite que o processo de engenharia de requisitos seja efetivamente utilizado para atingir seus objetivos sem a necessidade de complementações;
- Mínimo: de forma que seu foco é atingido com a utilização de poucos elementos.

Conforme os processos de requisitos tradicionais citados nos capítulos 3 e 4, o ProReqSE define um conjunto de atividades cujo objetivo é a compreensão, o refinamento, a documentação e a verificação das necessidades do cliente de forma a construir um software embarcado adequado. Para, isso foram definidas as seguintes atividades: definir visão, encontrar e resumir requisitos e detalhar requisitos.

As atividades deverão ser executadas na seqüência em que foram citadas acima, ou seja, primeiro deve ser definida a visão do sistema e posteriormente os requisitos devem ser resumidos e finalmente detalhados.

Estas atividades possuem artefatos de entrada e saída os quais foram denominados produtos de trabalho. Além disso, elas devem ser executadas pelos papéis definidos no processo.

Embora os papéis do ProReqSE exijam conhecimento específico de Engenharia de Requisitos e grande parte das empresas avaliadas no capítulo 06 não possuam equipes com este conhecimento, tal aspecto não será de grande relevância, visto que o ProReqSE foi elaborado de forma a apoiar todo o processo de engenharia de requisitos. Para isso foram definidos modelos de documentos (*templates*), diretrizes de trabalho, listas de verificações e conceitos específicos da engenharia de requisitos.

O primeiro ponto a ser destacado em relação ao processo definido é a abrangência de sua aplicação. Apesar da dificuldade na separação dos requisitos de hardware e software no domínio de sistemas embarcados, descritas na seção 4.4.3, o ProReqSE optou por manter a distinção entre estes dois componentes do sistema. Dessa forma, o ProReqSE abordará o processo de engenharia de requisitos de sistemas embarcados visando a construção de seu software. Dois são os motivos que levaram a esta decisão: a falta de

conhecimento, por parte da equipe desenvolvedora do processo, em relação à engenharia de hardware; as diferentes necessidades demandadas pelos dois processos de desenvolvimento – dificilmente uma técnica ou modelo para levantamento e especificação de requisitos de software seria aplicável ao hardware. No entanto, o ProReqSE não deixou de levar em consideração esta intrínseca ligação entre os dois componentes do sistema ao definir os elementos do processo.

A ferramenta EPF permite a publicação de um guia eletrônico do processo criado. Este guia visa facilitar de maneira significativa o entendimento e a institucionalização do processo nas empresas, visto que pode ser publicado em uma área pública, permitindo o acesso de todos os envolvidos. A Figura 20 ilustra a página inicial do Guia Eletrônico de Processos do ProReqSE:

Introdução ao ProReqSE

Introdução ao ProReqSE

Expand All Sections Collapse All Sections

Relationships

Contents

- Grupo de Disciplinas do ProReqSE
- Conceitos
- Papéis
- Produtos de Trabalho

Back to top

Main Description

Conceitos Papéis Produtos de Trabalho Grupo de Disciplinas do ProReqSE

O que é o ProReqSE?

O ProReqSE é um processo de engenharia de requisitos projetado especificamente para o desenvolvimento de sistemas embarcados por micro e pequenas empresas brasileiras. Ele foi desenvolvido com base no OpenUP, um processo iterativo que foi elaborado para aplicação em equipes pequenas e co-localizadas que buscam uma abordagem simplificada para o desenvolvimento de seus produtos. Como o ProReqSE foi projetado para micro e pequenas empresas brasileiras, este processo é:

- Customizável: de maneira que as empresas podem adequá-lo as suas necessidades de acordo com sua cultura e seu nível de formalização;
- Completo: permite que o processo de engenharia de requisitos seja efetivamente utilizado sem a necessidade de complementações;
- Mínimo: de forma que seu foco é atingido com a utilização de poucos elementos.

O ProReqSE define um conjunto de tarefas cujo objetivo é a compreensão, o refinamento, a documentação e a verificação das necessidades do cliente de forma a construir um software embarcado adequado. Logo ele deve compreender as seguintes tarefas:

- Definir Visão
- Encontrar e Resumir Requisitos
- Detalhar Requisitos

Os papéis definidos no ProReqSe são:

- Analista
- Arquiteto
- Desenvolvedor
- Envolvidos
- Gerente

E os produtos de trabalho definidos são:

- Documento de Visão
- Diagrama de Sequência de Cenários Operacionais
- Documento de Referência Cruzada
- Especificação de Caso de Uso
- Especificação Suplementar dos Requisitos
- Glossário
- Lista de Priorização das Necessidades
- Modelo de Caso de Uso

Figura 20 - Tela Inicial do ProReqSE

As seções a seguir apresentarão em detalhes os elementos que compõem o ProReqSE, bem como, as justificativas de sua utilização no domínio de sistemas embarcados e MPEs.

7.1.1 Papéis

Segundo as melhores práticas da ER descritas na seção 1.1.1.1 do capítulo 4, atribuir as atividades de ER a membros habilitados melhora a performance do desenvolvimento tornando-o mais previsível. Dessa forma, os papéis definidos no o ProReqSE levam em consideração as habilidades dos profissionais que devem assumi-los, além destas habilidades estarem diretamente relacionadas as atividades que cada papel executa. Os papéis definidos no ProReqSE são:

- **Analista:** representa os interesses dos clientes e dos usuários finais cujo objetivo é recolher as informações dos envolvidos para entender o problema a ser resolvido, capturar seus requisitos e definir suas prioridades.

O analista é o papel principal da ER, ele executa todas as atividades do processo e é o responsável pela elaboração da maioria dos produtos de trabalho. Dada sua importância no processo, foram listadas, a seguir, as habilidades e os conhecimentos exigidos do profissional que pretenda assumir este papel:

- Habilidade em identificar e entender os problemas e oportunidades;
 - Habilidade de articular as necessidades que estão associadas com os principais problemas a serem resolvidos ou com a oportunidade de ser realizada;
 - Habilidade de colaborar efetivamente com toda a equipe em sessões colaborativas de trabalho (como *workshops*, sessões de JAD e outras técnicas);
 - Boa capacidade de comunicação verbal e escrita;
 - Conhecimento dos domínios de negócio e de tecnologia ou a capacidade de absorver e compreender rapidamente tal informação.
- **Arquiteto:** responsável por fornecer apoio à tomada de decisões, balancear os interesses dos envolvidos, os riscos técnicos e assegurar que as decisões sejam comunicadas, validadas e seguidas de forma eficaz.

Visto que o arquiteto possui um papel secundário no processo, ele não é responsável pela execução de nenhuma atividade e nem pela elaboração dos produtos de trabalho.

- **Desenvolvedor:** responsável por fornecer apoio à tomada de decisões relacionadas à implementação. Assim como o arquiteto, não é responsável pela realização de nenhuma atividade e nem pela confecção de produtos de trabalho.
- **Envolvidos:** representam os grupos de envolvidos que precisam ser satisfeitos pelo projeto. É um papel que pode ser assumido por qualquer pessoal materialmente afetada pelo resultado do projeto. Da mesma maneira que o desenvolvedor e arquiteto não executa atividades específicas dentro do processo e nem elabora produtos de trabalho.
- **Gerente de Projetos:** conduz o planejamento do projeto, coordena as interações com os envolvidos e mantém a equipe de projeto focada em alcançar os objetivos do projeto.

7.1.2 Atividades

Conforme citado anteriormente, o ProReqSE foi dividido em quatro atividades, onde cada uma delas possui uma seqüência de tarefas a serem seguidas para atingir o objetivo da atividade.

Os itens seguintes detalham as atividades e suas respectivas tarefas:

- **Definir Visão:** definir a visão para o futuro sistema, ou seja, descrever o problema a ser resolvido e as características do sistema de acordo com as solicitações dos interessados. Vale ressaltar que neste momento não há separação entre hardware e software. A visão deve ser elaborada para o sistema como um todo.

O fluxo de execução da atividade Definir Visão é ilustrado na Figura 21.

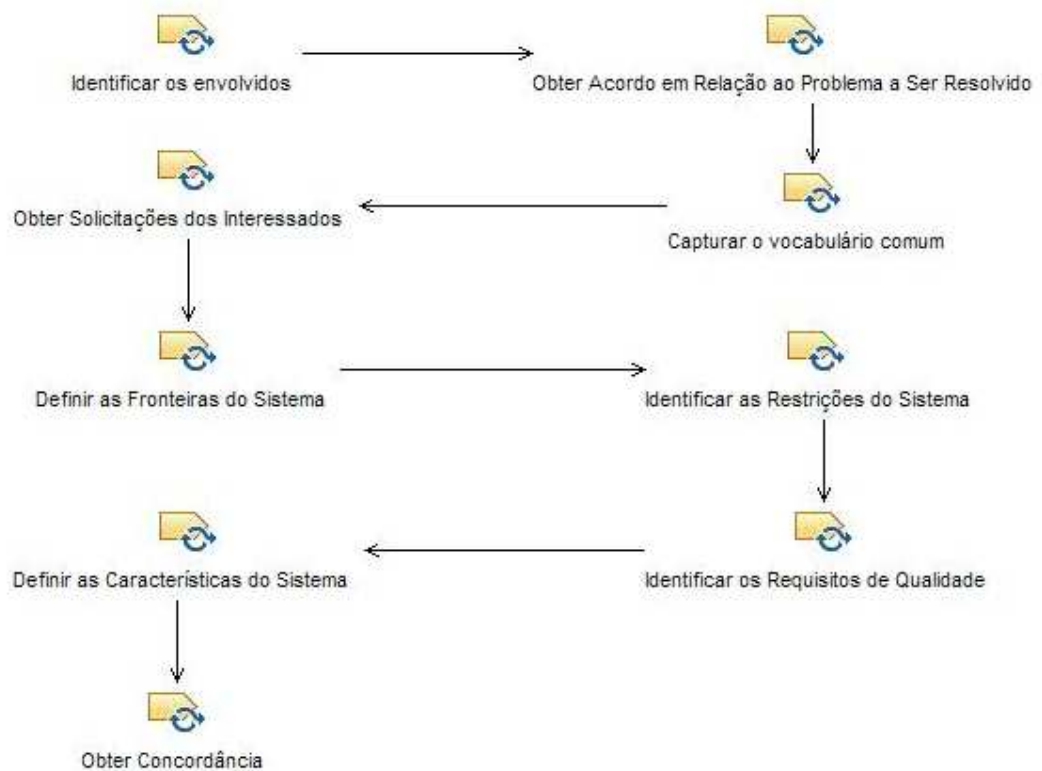


Figura 21 – Fluxo da Atividade Definir Visão

Conforme apresentado na Figura 21 esta atividade é subdividida nas seguintes tarefas:

- **Identificar os Envolvidos:** devem ser identificados todos os envolvidos com o desenvolvimento do sistema, como tomadores de decisão, clientes, usuários em potencial, parceiros, especialistas de domínio, analistas de mercado e outras partes interessadas. Devem ser determinados os perfis dos potenciais ou atuais usuários do sistema que se encaixem no papel de ator. Além disso, é importante que informações iniciais sobre os usuários-chave e seu ambiente sejam registradas no documento de visão.

Esta tarefa foi definida observando as características dos sistemas embarcados definida na seção 4.3.4 de que esse tipo de sistemas normalmente possui muitos envolvidos em seu desenvolvimento e que estes são os responsáveis por criar os requisitos do software. Para que o processo de requisitos atinja seu objetivo no desenvolvimento de software é necessário que estes profissionais estejam envolvidos no projeto desde a sua concepção.

- **Obter Acordo em Relação ao Problema a ser Resolvido:** para evitar qualquer mal-entendido durante o processo de desenvolvimento do software é importante que todos os envolvidos estejam de acordo com a definição do problema a ser resolvido, bem como com a terminologia a ser utilizada durante o projeto.

Após formular o problema esta informação deve ser registrada na seção correspondente do Documento de Visão, já a terminologia adotada deve ser documentada em um Glossário.

Esta é uma tarefa fundamental no processo de sistemas embarcados dada quantidade de interessados envolvidos no desenvolvimento e a complexidade do sistema a ser desenvolvido.

Técnicas sugeridas para executar esta etapa: *brainstorm*, *workshops*, entrevistas e sessões de JAD.

- **Capturar o Vocabulário Comum:** é de grande importância que se trabalhe junto aos interessados para criar um glossário que defina acrônimos, abreviações e termos técnicos e comerciais relevantes. Afinal, cada projeto tem sua terminologia específica e todos na equipe devem entendê-la bem, a fim de que a comunicação entre os envolvidos seja realizada de maneira efetiva. Vale ressaltar que este glossário deve ser continuamente refinado e expandido ao longo do projeto.

No domínio de sistemas embarcados é muito comum a presença de equipes multidisciplinares, portanto, primar por uma comunicação efetiva é de fundamental importância para o sucesso do projeto. Quanto maior a equipe e sua diversidade, maior será a necessidade de se manter um glossário atualizado do sistema.

- **Obter Solicitações dos Interessados:** deve ser utilizado o método mais apropriado para capturar informações. Foi desenvolvida uma diretriz para guiar esta tarefa, nesta diretriz são descritas várias técnicas, sendo que cada uma é aplicável a uma situação em particular ou a um certo tipo de interessado.

Dado o contexto de sistemas embarcados, o ideal, é o encontro presencial com todos os envolvidos, visto que isso reduz as chances da equipe de projeto entender as necessidades de maneira inadequada. Qualquer requisito capturado durante esta tarefa deve ser registrado na Lista de Priorização das Necessidades.

No contexto de sistemas embarcados nem sempre existe um cliente final participando do desenvolvimento do sistema, além do que muitas vezes os envolvidos possuem pouco conhecimento acerca do sistema desenvolvido. Para suprimir essas dificuldades, o ProReqSE utiliza a solução proposta por Puschig, descrita na seção 4.3.4, que sugere o envolvimento de especialistas de domínio. Associado a isso o ProReqSE sugere o uso de seções de JAD na execução dessa tarefa.

- **Definir as Fronteiras do Sistema:** Encontrar e definir a linha que divide a solução e o mundo real que a cerca. Essa tarefa consiste em identificar as interfaces bem como as informações de entrada e saída compartilhadas entre os usuários, sistemas, máquinas (ex: sensores, atuadores e hardware) e qualquer elementos do ambiente que interagem com o sistema (ex: pássaros e tempestades).

Esta é uma tarefa importante na definição dos requisitos de sistemas embarcados visto que este tipo de software normalmente possui um grande número de integrações. Além disso, a definição das fronteiras do sistema impacta diretamente na sua arquitetura.

Outra característica que deve ser considerada na execução desta tarefa é que os sistemas embarcados possuem tipos de interfaces diferentes dos sistemas tradicionais. A diretriz Encontrar e Resumir Atores e Caso de Uso auxilia a identificação das interfaces do sistema que são registradas nos modelos de caso de uso.

- **Identificar as Restrições do Sistema:** o objetivo desta tarefa é identificar todas as restrições que possam impactar no projeto. Os principais tipos de restrições dos sistemas embarcados são: políticas (normas, regulamentos, restrições de regulamentação e padrões legais), econômicas (orçamento, licenças), ambientais (execução em ambientes inóspitos), físicas (tamanho e peso), técnicas (plataformas, tecnologias – processador, memória), viabilidade (prazo, alocação de recursos) e sistema (compatibilidade de soluções, suporte a sistemas operacionais, ambientes, restrições de tempo real e gerenciamento de consumo de potência sem perda de desempenho).

É fundamental importância que essas restrições sejam identificadas de forma adequada, pois em tarefas posteriores elas serão refinadas em requisitos não-funcionais. Este aspecto é extremamente crítico em

sistemas embarcados dado que os requisitos não-funcionais são tratados tardiamente gerando impactos no produto final.

- **Identificar os requisitos de qualidade:** o objetivo desta tarefa é identificar todos os requisitos de qualidade que possam impactar no projeto. Aqui devem ser considerados aspectos como performance, usabilidade, confiabilidade, portabilidade, escalabilidade, manutenibilidade, reusabilidade, interoperabilidade, testabilidade, correção, consistência, compatibilidade, complexidade, i internacionalização (Fernandes 2005).

É fundamental importância que esses requisitos sejam identificados de forma adequada, pois em tarefas posteriores serão refinados em requisitos não-funcionais. Este aspecto é extremamente crítico em sistemas embarcados dado que os requisitos não-funcionais são tratados tardiamente gerando impactos no produto final.

- **Definir as Características do Sistema:** refinar as necessidades do envolvidos, indicando os recursos e as capacidades que o sistema deve possuir para atendê-las. Nessa tarefa as características são registradas no documento de visão.
- **Obter Concordância:** nesta tarefa os envolvidos devem revisar a visão do projeto a fim de garantir consentimento, determinar qualidade e identificar as mudanças requisitadas.

Como o desenvolvimento de sistemas embarcados normalmente é muito complexo e conta com a participação de diversos envolvidos, é importante que a visão do projeto seja revisada e acordada. Esta tarefa é apoiada pela Lista de Verificação de Visão.

- **Encontrar e Resumir Requisitos:** O propósito desta atividade é identificar e capturar requisitos funcionais e não-funcionais para o sistema. Estes requisitos formam a base da comunicação e concordância, entre os interessados e a equipe de desenvolvimento, a respeito do que o sistema deve fazer para atender as necessidades dos envolvidos. O seu objetivo é entender os requisitos em um alto nível de abstração para que seja possível determinar o escopo do sistema. Esta atividade pode ser realizada dividindo-se o sistema em componentes menores e construindo-se conexões entre eles. Posteriormente serão realizadas análises para detalhar os requisitos prioritários para implementação.

As tarefas da atividade Encontrar e Resumir Requisitos são ilustradas na Figura 22:

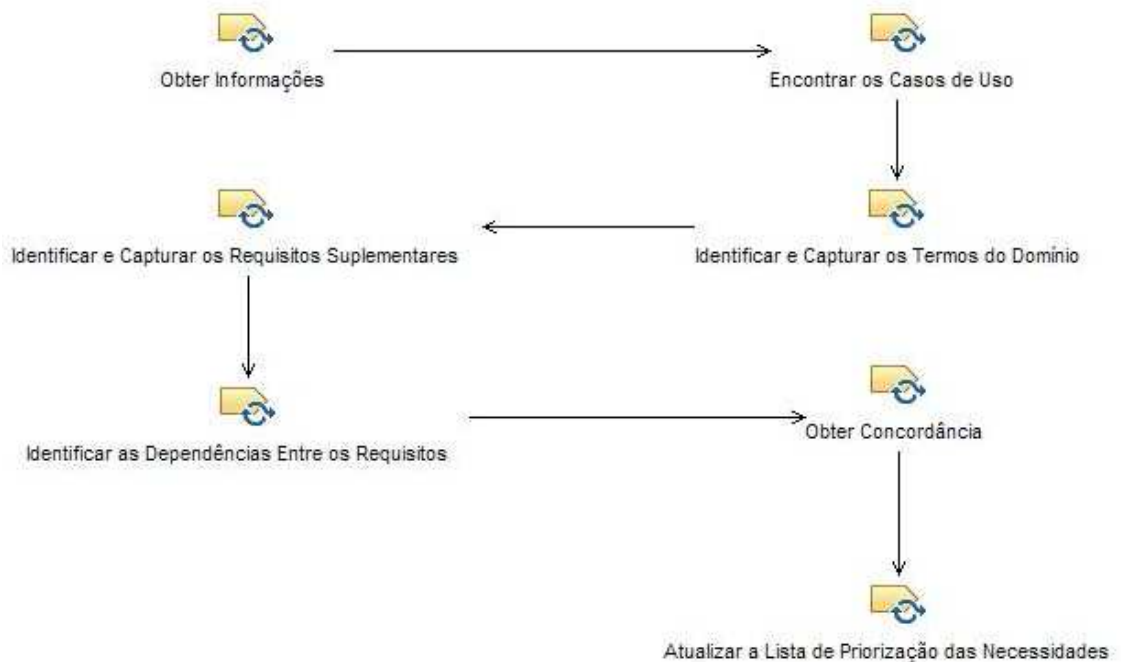


Figura 22 – Fluxo da Atividade Encontrar e Resumir Requisitos

Esta atividade é subdividida nas seguintes tarefas:

- **Obter Informações:** esta tarefa é executada com o apoio de diversas técnicas de obtenção de requisitos, como: *brainstorming*, JAD, questionários, *wokshops* dentre outras. Outra alternativa é a utilização de documentações de outros sistemas com características semelhantes ao que está sendo desenvolvido.

No desenvolvimento de sistemas embarcados, a participação de especialistas de domínio é de vital importância para definição de seus requisitos, uma vez que esse tipo de sistemas possui características específicas.

Conforme descrito do capítulo 4, a identificação e consulta de todas as origens dos requisitos e o envolvimento de clientes, usuários e especialistas durante a Engenharia de Requisitos melhoram respectivamente a cobertura dos requisitos e o entendimento das reais necessidades.

A partir deste momento é interessante que se comece a separar, explicitamente, o hardware do software. As atividades e tarefas seguintes

devem ser aplicadas apenas aos requisitos de software, ou seja, aqueles diretamente ligados ao aplicativo a ser desenvolvido.

- **Encontrar os Casos de Uso:** tomando como base as técnicas elaboradas por (Nars 2002), a principal estratégia para descobrir os casos de uso é o fato de se trabalhar baseado em funções e não em atores. Além de proporcionar maior flexibilidade, esta técnica permite que se encontre com maior habilidade todos os casos de uso funcionais do sistema. Para executar esta tarefa são sugeridos três passos: identificar as maiores funcionalidades do sistema, nomear cada caso de uso e descrevê-los brevemente.

No domínio de sistemas embarcados os tipos de funcionalidades mais comumente encontradas são: as de inicialização do sistema, monitoração de sensores e reporte de informações de status (Nars 2002). Muito cuidado deve ser tomado durante esta tarefa a fim de identificar apenas o que é necessário para o sistema em especificação. No domínio de sistemas embarcados é um grande desafio levantar apenas as funcionalidades do sistema a ser construído, dada a alta interatividade entre ele e os demais sistemas envolvidos.

- **Identificar e Capturar os Termos do Domínio:** o propósito desta tarefa é assegurar que os termos ambíguos ou específicos do domínio estão claramente definidos no glossário e que estão sendo utilizados de forma consistente.
- **Identificar e Capturar os Requisitos Suplementares:** o objetivo desta tarefa é obter os requisitos não-funcionais relevantes para o sistema, por meio da interação com os envolvidos. Outro instrumento de apoio são as restrições identificadas na atividade anterior, que são refinadas em requisitos suplementares.
- **Identificar as Dependências entre os Requisitos:** esta tarefa foi definida especificamente para o domínio de sistemas embarcados, onde os requisitos não-funcionais tendem a estar relacionados com mais de um requisito funcional. Esta relação pode ser de dependência, colaboração ou prejuízo. Buscando apoiar a especificação deste relacionamento, o ProReqSE sugere a utilização de um documento denominado Documento de Referência Cruzada. Além disso este produto de trabalho será extremamente útil na análise de impacto das mudanças no sistema. Esta

tarefa baseia-se nas melhores práticas da ER definidas na seção 1.1.1.1 e nos conceitos de rastreabilidade definidos no item 4.3.6.

- **Obter Concordância:** Esta tarefa visa conduzir revisões nos requisitos junto aos envolvidos para garantir consistência dos requisitos (funcionais e não-funcionais) com o documento de visão, determinar qualidade e identificar solicitações de mudanças. Essa tarefa é apoiada pelas listas de verificação modelo de caso de uso e de visão.
- **Atualizar a Lista Priorização de Necessidades:** Revisar os requisitos de acordo com as necessidades dos envolvidos para que os mesmos sejam priorizados na Lista de Priorização de Necessidades, conforme descrito na seção 1.1.1.1 do capítulo 04.
- **Detalhar Requisitos:** Esta atividade visa descrever os requisitos em detalhes, de acordo com sua priorização, para permitir a iniciação do desenvolvimento.

O fluxo da atividade Detalhar Requisitos é ilustrado na Figura 23:



Figura 23 – Fluxo da Atividade Detalhar Requisitos

- **Detalhar os Casos de Uso:** para executar esta tarefa sugere-se a utilização de um processo iterativo e progressivo de refinamento. O primeiro passo consiste em elaborar uma descrição breve do caso de uso, depois devem ser definidos e numerados os fluxos de eventos e, por fim, identificar as os requisitos não resolvidos, com o objetivo de esclarecê-los.

No item 7.1.3 foi definido um modelo para a especificação de caso de uso a fim de apoiar a execução desta tarefa. No entanto, caso se deseje elaborar um documento próprio, o detalhamento de caso de uso deve possuir no mínimo: o fluxo principal; como e quando o caso de uso começa e termina; quando o caso de uso irá interagir com os atores e o que será trocado entre eles; o que o sistema precisa fazer para executar o caso de uso; quando e como o caso de uso precisará de dados armazenados no sistema ou necessitará armazenar dados no sistema; eventos excepcionais; pré-condições e pós-condições.

As especificações de caso de uso devem ser detalhadas de acordo com sua priorização, a Diretriz Detalhar Caso de Uso apóia este trabalho. O nível de detalhe varia de acordo com as necessidades do projeto e a complexidade do caso de uso. A diretriz “Nível de Abstração dos Casos de Uso” auxilia a discussão sobre diferentes níveis de detalhe que podem ser aplicados aos casos de uso. As particularidades dos casos de uso são definidas na Especificação de Caso de Uso e nos Diagramas de Seqüência de Cenários Operacionais.

- **Detalhar Requisitos Suplementares:** Os requisitos não-funcionais devem ser detalhados no documento Especificação Suplementar de Requisitos, de acordo com a diretriz Requisitos Suplementares.

Conforme descrito no capítulo 4, requisitos não-funcionais são críticos para o desenvolvimento de sistemas embarcados. O ProReqSE, portanto, sugere a utilização de um método denominado PLanguage (Gilb 1989). Este método auxilia o levantamento de requisitos não-funcionais, evita a ocorrência de problemas como a falta de clareza e detalhamento de sua especificação e a confusão entre os requisitos funcionais e não-funcionais. Além disso, com a utilização da PLanguage os testes de requisitos não-funcionais podem ser executados com base na própria especificação, eliminando a necessidade de elaboração de um documento de testes. O documento de especificação suplementar irá agregar a tabela de especificação da PLanguage.

- **Detalhar Termos do Glossário:** Nesta tarefa todos os termos ambíguos ou específicos do domínio, utilizados nas especificações de caso de uso e no documento de especificação suplementar de requisitos, devem ser revisados e refinados conforme a necessidade.
- **Revisar Requisitos:** Esta atividade visa revisar todos os produtos de trabalho elaborados nas atividades e tarefas anteriores com o objetivo de validar o entendimento dos requisitos e garantir sua concordância com as expectativas dos envolvidos.

As tarefas da atividade Revisar Requisitos são ilustrados na Figura 24:



Figura 24 – Fluxo da Atividade Revisar Requisitos

- **Revisar Documentos:** Após a elaboração da visão do sistema e do detalhamento dos requisitos funcionais e não –funcionais o ProReqSE sugere a execução de uma revisão em todos os produtos de trabalho gerados, como, Documento de Visão, Glossário, Especificação de Caso de Uso, Especificação Suplementar, Documento de Referência Cruzada dentre outros. Essas verificações poderão ser executados, por meio de um dos três métodos de revisão descritos no capítulo 04: inspeção, *walkthrough* ou *revisão por pares*.

As listas de verificação irão guiar a execução das revisões. Os resultados dessa atividade devem ser registradas nestes documentos de forma que se possa construir uma base histórica dos erros cometidos em projetos passados.

- **Obter Concordância:** Conduzir uma revisão nos requisitos, especificação de caso de uso e especificação suplementar de requisitos, com os envolvidos relevantes a fim de garantir sua qualidade e consistência com o documento de visão e identificar solicitações de mudança.

7.1.3 Produtos de Trabalho

Conforme definido nos capítulos 03 e 04, os Produtos de Trabalho são informações produzidas, modificados ou utilizados pelas atividades do processo. Para facilitar o entendimento, a elaboração dos produtos de trabalho e a melhoria da qualidade da especificação foram definidos modelos (*templates*) destes documentos.

O ProReqSE sugere a utilização de poucos artefatos devido a seu direcionamento às MPEs e a cultura das equipes de desenvolvedoras de software embarcado. No entanto, estes artefatos podem ser acrescidos, suprimidos ou customizados de acordo com a necessidade de cada projeto.

Os modelos de documentos dos produtos de trabalho definidos ou customizados pelo ProReqSE, para o domínio específico de sistemas embarcados, são disponibilizados nos APÊNDICES E, F e G.

O ProReqSE fornece os seguintes produtos de trabalho:

- **Documento de Visão:** Este artefato fornece uma base de alto nível, às vezes contratual, para os requisitos técnicos mais detalhados que são visíveis para os envolvidos. Captura a essência do sistema descrevendo as restrições do projeto

e os requisitos de alto nível que dão ao leitor uma visão geral do sistema de uma perspectiva dos requisitos comportamentais.

- **Especificação de Caso de Uso:** Este artefato captura a seqüência das ações executadas por um sistema que tenham um resultado de valor observável para aqueles que interagem com o sistema. O principal diferencial da especificação de caso de uso utilizada pelo ProReqSE é a definição de tipos de atores específicos do domínio de sistemas embarcados, descritos na tarefa Definir as Fronteiras do Sistema.
- **Especificação Suplementar:** Este artefato captura e detalha todos os requisitos do sistema que não são especificados nos casos de uso, incluindo requisitos de atributos de qualidade e requisitos funcionais globais. O maior diferencial da Especificação Suplementar do ProReqSE é a utilização da tabela de especificação da PLanguage.
- **Glossário:** O objetivo do glossário é fornecer um vocabulário comum acordado por todos os envolvidos. Ajuda pessoas de diferentes grupos funcionais a alcançar uma compreensão mútua do sistema. O objetivo não é registrar todos os termos possíveis, mas somente aqueles que não estão claros, são ambíguos ou pertencem a domínios específicos.
- **Lista de Priorização das Necessidades:** O objetivo deste artefato é priorizar as necessidades dos envolvidos a fim de guiar a seqüência do detalhamento, projeto e implementação do sistema.
- **Modelo de Caso de Uso:** Este artefato captura um modelo das funções desejadas do sistema e de seu ambiente, e serve como um contrato entre o cliente e os desenvolvedores. O principal diferencial do modelo de caso de uso utilizado pelo ProReqSE é a definição de tipos de atores específicos do domínio de sistemas embarcados.
- **Documento de Referência Cruzada:** A finalidade deste documento é manter a rastreabilidade e a concorrência (dependência, prejuízo ou colaboração) entre os elementos do projeto. O ProReqSE sugere que seja feita a rastreabilidade entre necessidades, características, requisitos funcionais, requisitos não-funcionais. Contudo, outros elementos do projeto podem ser rastreados.
- **Diagrama de Seqüência de Cenários Operacionais:** Este documento visa representar a seqüência temporal da execução de cada um dos cenários dos casos de uso. De maneira que a concorrência entre os fluxos possa ser tratada a

fim de definir a correta seqüência a ser obedecida durante a execução. Por exemplo: identificar qual exceção deve ser priorizada caso duas exceções sejam lançadas no mesmo momento.

7.1.4 Diretrizes

As diretrizes são guias que buscam facilitar a execução das tarefas que compõe as atividades definidas do ProReqSE e conseqüente a elaboração de seus produtos de trabalho. Estes itens são de vital importância para o processo dado o contexto das equipes desenvolvedoras de sistemas embarcados.

As diretrizes disponibilizadas no ProReqSE são:

- Armadilhas de Requisitos;
- Detalhar Casos de Uso;
- Encontrar Caso de Uso;
- Encontrar e Resumir Atores e Caso de Uso;
- Escrever Requisitos Adequadamente;
- Formato dos Casos de Uso;
- Modelo de Caso de Uso;
- Realizar a Revisão dos Requisitos;
- Requisitos Suplementares;
- Técnicas para Obtenção de Requisitos.

7.1.5 Listas de Verificação

Conforme definido no capítulo 04, as listas de verificação visam garantir a qualidade e a completude dos produtos gerados em cada uma das atividades do processo de engenharia de requisitos para sistemas embarcados, bem como, a alta satisfação do cliente. O ProReqSE utiliza as Listas de Verificação definidas a seguir na atividade Revisar Requisitos:

- Lista de Verificação de Ator;
- Lista de Verificação de Bons Requisitos;
- Lista de Verificação de Casos de Uso;
- Lista de Verificação de Modelos de Caso de Uso – Disponível no APÊNDICE C;

- Lista de Verificação de Requisitos Suplementares – Disponível no APÊNDICE D;
- Lista de Verificação de Visão – Disponível no APÊNDICE B;

7.1.6 Conceitos

O objetivo dos conceitos é definir de forma detalhada termos e técnicas freqüentemente utilizados na Engenharia de Requisitos. Da mesma forma que as diretrizes, este elemento do processo auxilia as equipes de desenvolvimento de sistemas embarcados, uma vez que estas não possuem conhecimentos específicos de Engenharia de Requisitos.

Os conceitos fornecidos pelo ProReqSE são:

- Ator;
- Atributos de Requisitos;
- PLanguage;
- Característica;
- Caso de Uso;
- Modelo de Caso de Uso;
- Rastreabilidade;
- Requisitos;
- Requisitos Arquiteturalmente Significantes;
- Requisitos Suplementares.

7.2 Comparação do ProReqSE com Outros Processos

Conforme foi observado nos capítulos 03, os processos de desenvolvimento de software embarcado se assemelham muito aos processos tradicionais. Apesar destes processos serem específicos para o domínio de sistemas embarcados eles dão maior enfoque ao projeto do software. A Tabela 24 descreve as diferenças básicas entre os processos de sistemas embarcados apresentados na literatura e o ProReqSE.

Tabela 24 - Comparativo das Características de ER em Processos de Desenvolvimento de Sistemas Embarcados

Característica do Processo de ER	RUP SE	IPProcess	UPES	ProReqSE
				S
				S

	S
	S
	S
	S

O comparativo do ProReqSE com outros processos de desenvolvimento de sistemas embarcados evidencia os valores que foram agregados no processo elaborado. A separação do tratamento dos requisitos e do projeto de software e a inserção de tarefas, atividades, papéis, conceitos, diretrizes e listas de verificação específicas para este domínio são características que reforçam essa afirmação.

7.3 Considerações Finais

As atividades do processo proposto devem ser executadas durante as fases do desenvolvimento do software, no entanto, nada impede que alguma atividade ou tarefa seja executada mais de uma vez ou seja quebrada com o intuito de adequar os artefatos gerados. É importante que os produtos gerados a partir do processo de engenharia de requisitos dêem subsídios à execução dos processos subseqüentes, de forma que o produto final atinja seus objetivos.

Neste capítulo foi descrito o processo de engenharia de requisitos para sistemas embarcados (ProReqSE) criado a partir da avaliação, detalhada no capítulo 06, do referencial teórico, exposto nos capítulos 02, 03 e 04 e do conhecimento prático na área de Engenharia de Requisitos.

O capítulo seguinte apresenta as conclusões e trabalhos futuros relativos ao estudo desenvolvido.

8. CONSIDERAÇÕES FINAIS DO TRABALHO

8.1 Conclusão

No presente trabalho foi apresentada uma proposta de processo de engenharia de requisitos para sistemas embarcados. Esta proposta foi elaborada com base nos resultados de uma avaliação do processo de desenvolvimento de micro e pequenas empresas desenvolvedoras de sistemas embarcados e estudos sobre processos de desenvolvimento, processo de requisitos, software embarcado e GQM.

A linha de pesquisas de sistemas embarcados que vem crescendo muito nos últimos anos devido à evolução, não apenas dos microprocessadores, mas também de outros componentes de hardware. Conseqüentemente, a quantidade de produtos no mercado compostos de sistemas embarcados aumentou de forma significativa. No entanto, o software utilizado nesses produtos não tem conseguido acompanhar essa evolução.

Inúmeras dificuldades foram encontradas durante o desenvolvimento do trabalho, dentre elas a escassez não só de profissionais da área de ciência da computação com conhecimento em software embarcado, mas também de material de apoio à pesquisa sobre processos de desenvolvimento e processos de requisitos para esse domínio de sistemas. Contudo, esses obstáculos tornaram-se incentivos à realização do trabalho.

Outro empecilho encontrado foi a carência de micro e pequenas empresas desenvolvedoras de software embarcado que estivessem dispostas a participar da pesquisa. Além disso, o fato dos pesquisadores envolvidos neste estudo não vivenciarem o contexto do desenvolvimento de sistemas embarcados dificultou a elaboração do programa de medições utilizado na avaliação. O nível de detalhe das questões, os termos a serem utilizados no questionário e a definição das métricas de cada ponto de verificação possivelmente não tenham sido definidos de forma que pudessem avaliar precisamente as deficiências e os pontos fortes do processo de requisitos das empresas participantes.

Mesmo tendo contatado três empresas que confirmaram a sua participação na pesquisa, apenas uma delas respondeu ao questionário. Isso prejudicou a pesquisa pela falta amostras mas serviu como prova de conceito inicial para o GQM idealizado. Apesar disso, a junção dos resultados obtidos na avaliação com as características do desenvolvimento de software embarcado identificadas nas pesquisa serviram como subsidio à elaboração do ProReqSE.

O perfil dos profissionais envolvidos no desenvolvimento de software embarcado geralmente é voltado para a área de engenharia, esta é uma característica que dificultou a

elaboração dos papéis do ProReqSE, pois as habilidades dos envolvidos não condizem com as habilidades requeridas. Esse problema foi solucionado com a elaboração de diretrizes e a definição dos conceitos que dão suporte a execução das atividades do ProReqSE.

Por fim, a combinação do uso do GQM na avaliação do processo de requisitos existente nas MPEs desenvolvedoras de software e a utilização de conceitos da literatura de sistemas embarcados demonstrou uma boa aplicabilidade na definição do ProReqSE.

8.2 Trabalhos Futuros

Este projeto foi pioneiro na Universidade Católica de Brasília na área de sistemas embarcados. Como o trabalho abordou apenas o processo de Engenharia de Requisitos de Micro e Pequenas Empresas desenvolvedoras de software, existem diversas oportunidades para dar continuidade a essa linha de pesquisa:

- Ampliar a pesquisa abordando outros processos do desenvolvimento de software.
- Refinar a avaliação elaborando outras questões, pontos de verificação e métricas.
- Aplicar a avaliação coletando informações em uma maior quantidade de empresas com mais respondentes.
- Refinar o ProReqSE com a sugestão de novas adequações.
- Elaborar um processo completo de desenvolvimento de software embarcado.
- Implantar o processo em uma ou mais MPEs.
- Obter e armazenar indicadores que avaliem a eficiência do ProReqSE e sugira melhorias.
- Conseguir parceria de empresas ou instituições que trabalhem com o desenvolvimento de software embarcado.
- Adequar o ProReqSE de maneira que possa estar aderente a modelos de melhoria de processo de software.

REFERÊNCIAS

- Acuña, A., M. Ferre, M. López, e L. Mate. "The Software Process: Modeling, Evaluation and Improvement." *World Scientific Publishing Company*, 2000.
- Anquetil, N., K. M. Oliveira, e K. D. Souza. "Uso do GQM para avaliar implantação de processo de manutenção de software." *IV Simpósio Brasileiro de Qualidade de Software*, 2005.
- Barbiero, A. A. *Ambiente de Suporte ao Projeto de Sistemas Embarcados*. Curitiba: Universidade Federal do Paraná, 2006.
- Barreto, R.S. *Sistemas Embarcados: o novo boom da Informática?* Sociedade Brasileira de Computação, 2006.
- Barros, E., G. Esmeraldo, e A. L. M. Silva. "Uma Metodologia de Desenvolvimento Baseada em Componentes Aplicada à Modelagem de Sistemas Embarcados." *Workshop de Desenvolvimento Baseado em Componentes (WDBC)*, 2006.
- Barros, E., J. Bione, M. Lima, e F. Santos. "IpProcess: A Development Process for Soft Ip-Core with Prototyping in FPGA." *Forum on Specification & Design Language*, 2005.
- Basili, V., e H. Rombach. *Goal Question Metric Paradim. Encyclopedia of Software Engineering*. New York: John Riley and Sons, 1994.
- Berghout, E., e R. Solligen. *The Goal Question Metric Method: A practical Guide for Quality Improvement of Software Development*. London: McGraw Hill, 1999.
- Calsalvara, A., C. A. F. Machado, S. S. Reinehr, e R. C. Burnett. "Norma NBR ISO/IEC 12207." 2000.
- Cantor, M. "Rational Unified Process for System Engineering - Introdução RUPSE." *The Rational Edge*, 2003.
- Cantor, M. "Rational Unified Process for Sytem Enginerring - Análise e Design de Requisitos." *The Rational Edge*, 2003.
- Carro, L. and Wagner, F.R. *Sistemas Computacionais Embarcados*. Vol. 56. Academic Press, 2002.
- Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 1999.
- Cysneiros, L. M. "Requisitos Não Funcionais: Da Elicitação ao Modelo Conceitual." *PUC/RJ*, 2001.
- Dórias, E. S. *Replicação de Estudos Empíricos em Engenharia de Software*. Edição: Matemáticas e Computação Instituto de Ciências. ICMC - USP, 2001.
- Farias, T. M.M. "Aplicação de Padrões ao Processo de Desenvolvimento de Software RUP." *Trabalho de Conclusão de Curso - Universidade de Pernambuco*, 2006.

- Feiler, P. H., e W. S. Humphrey. "Software process development and enactment: concepts and definitions." *Second International Conference on the Continuous Software Process Improvement* (IEEE Computer Society), 1993: 28 - 40.
- Fernandes, D. B. *Análise de Sistemas Orientada ao Sucesso: Por que os projetos atrasam?* Ciência Moderna, 2005.
- Foudation, Eclipse. *Eclipse Process Framework*. <http://www.eclipse.org/epf> (acesso em 01 de 05 de 2008).
- Gastaldo, D.L. e Midorikawa, E.T. *Processo de Engenharia de Requisitos Aplicado a Requisitos Não-Funcionais de Desempenho – Um Estudo de Caso*. 2003.
- Gilb, T. "A Planning Language." Edição: ACM Press. *International Conference on APL*. 1989. 169-177.
- Gladcheff, A. P., R. Sanches, e D. M. da Silva. "Um Instrumento de Avaliação de Qualidade de Software Educacional: como elaborá-lo." *Simpósio Brasileiro de Engenharia de Software - VIII Workshop de Qualidade de Software*, 2001.
- Goethert, W., e M. Fischer. "Deriving Enterprise-Based Measure Using the Balanced Scorecard and Goal-Driven Measurement Techniques." *Software Engineering Institute*, 2003.
- Gonçalves, J. E. L. "As empresas são grandes coleções de processos." *Revista de Administração de Empresas*, Jan/Mar 2000: 6-19.
- Graaf, B. and Lormans, M. and Toetenel, H. "Embedded Software Engeneering: The State of the Practice." *IEEE Software*, 2003: 61-69.
- Hauck, J. C. "Modelagem e Aplicação de um Processo de Software em uma Microempresa." Relatório Técnico, UNIVALI, São José, 2002.
- Hauck, J. C. R. "Desenvolvimento de um Sistema de Software para Gerência de Manuais de Processos de Software em Micro e Pequenas Empresas." *Trabalho de Conclusão de Curso*, 2004.
- Henzinger, T. e Sifakis, J. "The Discipline Of Embedded Systems Design." *IEEE Computer Magazine* 40 (2007): 32-40.
- Hewett, R. e Seker, R. "A Risk Assessment Model of Embedded Software Systems." *IEEE International Conference on Man and Cybernetics Systems* 4 (2005): 3238- 3243.
- Hofmann, H.F. e Lehner, F. *Requirements Engineering as a Success Factor in Software Projects*. Vol. 18. 4 vols. 2001.
- IEEE. *SWEBOK - A Project of the Software Engineering Coordinating Committee*. 2001.
- Jacobson, I., G. Boock, e J. Rumbaugh. *The Unified Software Development Process*. Addison - Wesley, 1999.
- Kondo, M.N.S., Silva, J.R., Hira, A.Y. e Zuffo, M.K. "Estudo de Requisitos do Software Embarcado no Segmento da Telemedicina." *Escola Politécnica da USP*, 2006.

Kopanas, V., V. Sylaidis, e I. Nakakis. "GQM-based improvement of embedded, real-time software development practices." *Eighth IEEE International Workshop on incorporating Computer Aided Software Engineering*, 1997: 105-115.

Kruchten, P. *Introdução ao RUP - Rational Unified Process*. Rio de Janeiro: Ciência Moderna, 2003.

Laboratory, Software Measurement. *GQM Method*. <http://ivs.cs.uni-magderburg.de/sw-eng/us/java/GQM/link2.shtml> (acesso em 10 de 2007).

Larman, C. *Utilizando UML e Padrões*. São Paulo: Bookman, 2004.

Lattemann, F. e Lehmann, E. A. "Methodological Approach to the Requirement Specification of Embedded Systems." *Proceedings of the 1st International Conference on Formal Engineering Methods (ICFEM'97)*, 1997: 183.

Lee, E.A. *Embedded Software*. Vol. Vol 56. Academic Press, 2002.

Lee, E.A. "What's Ahead for Embedded Software?" *IEEE Computer*, 2000: 19-26.

Lee, S., Ko, H., Jo, D., Jeong, J., Kim, K. "Reusable SW Requirements Development Process: Embedded SW Industry Experiences." *Proceedings of the 2007 Australian Software Engineering Conference (ASWEC'07)*, 2007: 147-158.

Marwedel, P. *Embedded System Design*. Springer, 2003.

Mcchesney, R. T. "Toward a Classification Scheme for Software Process Modelling Approaches." *Information and Software Technology*, 1995.

MCT, Ministério da Ciência e Tecnologia. "Qualidade e Produtividade no Setor de Software Brasileiro." *Comunicação Pessoal*, 2002.

Nars, E., McDermid, J. e Bernat, G. "Eliciting and Specifying with Use Cases for Embedded Systems." *Proceedings of the Seventh International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2002)*, 2002: 350-357.

Oliveira, M. S. "Orientações Metodológicas para Monografia Latu Sensu."

OMG, Object Management Group. "Software Process Engineering Metamodel Specification." 2005.

Pimentel, A. R. "Uma Abordagem para Projeto de Software Orientado a Objetos Baseado na Teoria de Axiomático." *Tese de Doutorado - Universidade Tecnológica Federal do Paraná*, Maio 2007.

Pradhan, D. K. *Fault-Tolerant System Design*. New Jersey: Prentice Hall, 1996.

Pressman, R. S. *Engenharia de Software*. 5ª edição. Mc Graw Hill, 2006.

Puschnig, A. e Kolagari, R.T. "Requirements Engineering in the Development of Innovative Automotive Embedded Systems." *Proceedings of the 12th International Requirements Engineering Conference (RE'04)*, 2004: 328-333.

Ramos, C. "Avaliação de Sistemas Legados." *Dissertação de Mestrado - Universidade Católica de Brasília - UCB*, 2004.

Riccobene, E., P. Scandurra, A. Rosti, e S. Bocchio. "Designing a Unified Process for Embedded Systems." Edição: IEEE Computer Society. *Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software MOMPES07*, Março 2007: 77 - 90.

Robertson, J. "Eureka! Why Analysts Should Invent Requirements." *IEEE Software*, 2002: 20-22.

RUP. "Site RUP." Wthreex. 2001.

http://www.wthreex.com/rup/process/workflow/requirem/co_req.htm (acesso em 02 de 02 de 2008).

Sangiovanni-Vincentelli, A., Carloni, L., De Bernardinis, F., and Sgroi, M. *Benefits and challenges for platform-based design*. 2004.

Silva, C.A.L.B., Silva, M.A.L.R. e Costa, O.L.P. "Implantação de Melhoria no Processo "Engenharia de Requisitos" na Empresa Fórmula Informática." *VI Simpósio Internacional de Melhoria de Processos de Software (SIMPROS 04)*, 2004.

Softex. *Melhoria de Processo de Software Brasileiro - Guia Geral Versão 1.2*.

<http://www.softex.br/mpsbr/> (in portuguese) (acesso em 15 de 11 de 2007).

Sommerville, I. e Kotonya, G. *Requirements Engineering*. Wiley, 1998.

Tennenhouse, D. *Proactive Computing*. 5ª edição. Vol. 3. 2000.

Wangenheim, C. G. V., e G. Ruhe. "Análise de Custo Benefício de Mensuração Baseada em GQM – Um Estudo de Caso Replicado." *Proceedings of X Conferência Internacional de Qualidade de Software*, 1999.

Weber, S. "Um Estudo de Caso em uma Microempresa de Software para o Desenvolvimento de um Modelo de Processo de Software." *Universidade Federal de Florianópolis - UFSC*, 2002.

Wolf, W. *Computers as Components: Principles of Embedded Computing System Design*. McGraw-Hill, 2001.

Yamaura, T., Miyazaki, H. e Onoma, A.K. "A New Defining Approach for Software Requirement Specifications." *Proceedings of the IEEE Workshop on Software Technologies for Future Embedded Systems (WSTFES'03)*, 2003: 13-16.

APÊNDICE A – RESPOSTA DO QUESTIONÁRIO DE AVALIAÇÃO DOS PROCESSOS DE DESENVOLVIMENTO DE SISTEMAS EMBARCADOS

Questionário de Avaliação dos Processos de Desenvolvimento de Sistemas Embarcados

Equipe Responsável pela Avaliação:

Cláudia de Oliveira Melo

Daniele Erica Damke

Patrícia Freitas de Moraes

Equipe Avaliada:

Latenter Criptografia e Segurança Digital

Thiago de Castro Martins

Engenheiro de Desenvolvimento

1. Contextualização - A empresa		
Questões		Respostas
1.1	Qual o domínio de negócio da empresa?	<i>Desenvolvimento de hardware criptográfico</i>
1.2	Há quanto tempo a empresa atua neste domínio de negócio?	<i>3 anos</i>
1.3	Qual o número de funcionários da empresa?	<i>6</i>
Dúvidas	<i>[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]</i>	

2. Contextualização - O software		
Questões		Respostas
2.1	O software embarcado desenvolvido pela empresa é...	<i>(2) Um produto da empresa (cliente interno).</i>
2.2	O software embarcado desenvolvido pela empresa possui integração com outros sistemas?	<i>(3) Possui integração com sistemas da própria empresa e de</i>

		<i>outros fabricantes.</i>
2.3	<i>O software embarcado desenvolvido pela empresa deve executar em diferentes plataformas?</i>	<i>(0) O software é dedicado a uma plataforma específica da empresa.</i>
Dúvidas	<i>[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]</i>	

3. Contextualização - A Equipe de Requisitos		
Questões		Respostas
3.1	<i>No caso do software ser um componente de um produto da própria empresa ou de outro fabricante, existe uma equipe separada para o levantamento dos requisitos do software?</i>	<i>(2) Não existe separação entre as equipes. Mas os levantamentos do produto e do software são realizados em momentos distintos e por processos distintos.</i>
3.2	<i>Caso exista uma equipe separada para os requisitos do software, qual o seu tamanho, em relação ao tamanho total da equipe responsável pelo projeto?</i>	<i>(0) Não existe uma equipe separada.</i>
3.2	<i>Caso exista uma equipe separada para os requisitos do software, seus membros são dedicados a esta atividade?</i>	<i>(0) Não existe uma equipe separada.</i>
3.4	<i>Qual a formação acadêmica dos membros da equipe de requisitos (mesmo que não exista uma equipe separada para tal)?</i>	<i>(3) Os membros da equipe possuem formação em Ciência da Computação (ou áreas afins), e possuem conhecimento superficial da Engenharia de Requisitos.</i>
3.5	<i>Qual o nível de conhecimento da equipe de requisitos no que diz respeito ao negócio do software a ser desenvolvido?</i>	<i>(2) A equipe de requisitos tem conhecimentos aprofundados do negócio.</i>
Dúvidas	<i>[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]</i>	

4. O entendimento dos requisitos (funcionais e não-funcionais) é obtido junto aos fornecedores de requisitos?		
Questões		Respostas
4.1	<i>Qual o nível de entendimento dos requisitos?</i>	<i>(2) Os requisitos são recebidos já levantados.</i>
4.2	<i>Qual o nível de especialização da equipe de levantamento de requisitos nesta atividade?</i>	<i>(3) Os requisitos são levantados pelos arquitetos ou projetistas.</i>
4.3	<i>Qual o nível de detalhe do levantamento de requisitos?</i>	<i>(3) As necessidades do cliente, as restrições do sistema e as</i>

	<i>integrações com outros sistemas/dispositivos são identificadas.</i>
Dúvidas	<i>[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]</i>

5. As necessidades, expectativas e restrições do cliente, tanto do produto quanto de suas interfaces, são identificadas e registradas?	
Questões	
Respostas	
5.1	<i>Qual o nível de formalização dos requisitos?</i>
	<i>(3) Os requisitos são acordados e registrados em documentos estruturados (ex: atas de reunião).</i>
5.2	<i>Qual o nível de formalização do detalhamento do levantamento/especificação de requisitos?</i>
	<i>(3) As necessidades, expectativas e restrições do cliente são levantadas e registradas em atas de reunião.</i>
Dúvidas	<i>[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]</i>

6. Os requisitos de software são aprovados?	
Questões	
Respostas	
6.1	<i>Qual o nível de formalização dos requisitos?</i>
	<i>(4) Os requisitos são acordados e registrados formalmente (ex: documento de especificação de requisitos).</i>
Dúvidas	<i>[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]</i>

7. Um conjunto definido de requisitos do cliente é especificado a partir das necessidades, expectativas e restrições identificadas?	
Questões	
Respostas	
7.1	<i>Qual o nível de refinamento dos requisitos funcionais?</i>
	<i>(2) As características são diretamente levantadas e refinadas em nível de requisitos.</i>
Dúvidas	<i>[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]</i>

8. Um conjunto de requisitos funcionais e não-funcionais do produto e dos componentes do produto que descrevem a solução do problema a ser resolvido, é definido e mantido a partir dos requisitos do cliente?

Questões		Respostas
8.1	Qual o nível de entendimento dos requisitos não-funcionais?	(2) Os requisitos não-funcionais são identificados e são definidos a partir dos requisitos funcionais identificados.
Dúvidas	[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]	

9. É utilizado algum método, modelo ou linguagem para especificação dos requisitos (funcionais e não-funcionais)?		
Questões		Respostas
9.1	Qual o nível de utilização de métodos/modelos no levantamento/especificação dos requisitos (funcionais e não-funcionais)?	(1) Os requisitos funcionais e não-funcionais são levantados/especificados sem utilização de técnicas auxiliares.
Dúvidas	[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]	

10. É estabelecida a rastreabilidade bidirecional entre os requisitos e os produtos de trabalho? Em caso positivo, esta rastreabilidade é mantida?		
Questões		Respostas
10.1	Qual o nível de rastreabilidade dos requisitos?	(1) A rastreabilidade entre os requisitos de trabalho é elaborada.
Dúvidas	[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]	

11. Interfaces internas e externas do produto e de cada componente do produto são definidas?		
Questões		Respostas
11.1	Qual o nível de identificação de interfaces?	(3) Os atores do sistema são definidos e registrados em documentos estruturados (Ex. Ata de reunião).
Dúvidas	[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]	

12. Os requisitos são analisados para assegurar que sejam necessários, corretos, testáveis e suficientes para balancear as necessidades dos interessados com as restrições existentes?		
Questões		Respostas

12.1	Qual o nível de análise de erros nos requisitos?	(1) Os requisitos são testados pela equipe de desenvolvimento e não são validados pelo cliente.
12.2	Qual o nível de correteude dos testes de requisitos?	[Favor informar a opção que mais se aplica à sua realidade]
Dúvidas	Acredito que haja algum erro nas alternativas de resposta da questão 13.2	

13. É considerada e avaliada a dependência entre os requisitos?		
Questões		Respostas
13.1	Qual o nível de análise da dependência entre os requisitos?	(1) A dependência entre os requisitos funcionais é avaliada.
Dúvidas	[Favor digitar aqui suas dúvidas e comentários em relação a questão]	

14. É considerada e validada a viabilidade dos itens de software atenderem seus requisitos alocados?		
Questões		Respostas
14.1	Qual o nível de análise da viabilidade dos requisitos?	(1) A viabilidade dos requisitos funcionais é avaliada.
Dúvidas	[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]	

15. É considerada e avaliada a viabilidade da operação e da manutenção do sistema?		
Questões		Respostas
15.1	Qual o nível de análise da viabilidade de operação e manutenção dos sistemas?	(1) A viabilidade de operação do sistema é avaliada.
Dúvidas	Por "manutenção" deve-se entender a manutenção do produto final (hardware + software) ou a atualização do software embarcado?	

16. São feitas revisões nos planos e produtos de trabalho do projeto visando identificar e corrigir inconsistências em relação aos requisitos?		
Questões		Respostas
16.1	Qual o nível de análise de revisões e correções dos produtos de trabalho?	(1) São realizadas revisões e correções dos planos e produtos de trabalho durante o processo de desenvolvimento do sistema.

Dúvidas	<i>[Favor digitar aqui suas dúvidas e comentários em relação a questão, caso existam]</i>
----------------	-------------------------------------------------------------------------------------------

17. As mudanças nos requisitos são gerenciadas ao longo do tempo?		
	Questões	Respostas
17.1	<i>Qual o nível de análise de manutenções nos requisitos funcionais e não-funcionais?</i>	<i>(2) Os requisitos funcionais e não-funcionais são mantidos ao longo do tempo.</i>
Dúvidas	<i>As manutenções mais freqüentes no meu negócio consistem na geração de versões especializadas do produto. Neste caso, evidentemente os requisitos são passíveis de sofrerem alterações. No entanto, estes novos requisitos NÃO substituem os originais. Em termos de um sistema de controle de revisões, isso seria o equivalente a um "branch".</i>	

Comentários
<i>[Favor digitar aqui seus comentários, críticas e sugestões em relação ao questionário]</i>

APÊNDICE B – LISTA DE VERIFICAÇÃO I

PROCESSO DE DESENVOLVIMENTO DE SISTEMAS
EMBARCADOS

Logotipo da Empresa

Lista de Verificação do Documento de Visão

<Sigla do Projeto> - <Nome do Projeto>		
Lista de Verificação do Documento de Visão: <Nome do Documento>	Versão <X>	Data: 18/3/2008

1. IDENTIFICAÇÃO

[Descrever informações gerais sobre a revisão técnica sendo realizada:

- Projeto: Sigla e nome do projeto ao qual o(s) produto(s) sendo revisados pertencem;*
- Nome do Revisor: Nome da pessoa responsável pela revisão;*
- Data da Revisão: Data da realização da revisão;*
- Tempo de duração da revisão: Tempo gasto, em horas e minutos, com a atividade de revisão técnica do produto.]*

Projeto	<Sigla do Projeto> - <Nome do Projeto>		
Nome do Revisor			
Data da Revisão	<dd/MM/AAAA>	Tempo de Duração da Revisão	<X> horas e <X> minutos

2. OBJETIVO

[Descrever o objetivo da revisão e listar os produtos a serem revisados.]

Propósito da Revisão	
Revisar o documento Visão identificando desvios relativos aos requisitos de alto nível que representam o escopo do sistema e a rastreabilidade entre esses requisitos.	
Produtos a serem revisados	Versão
<Visão>	<X>

3. ITENS DE VERIFICAÇÃO

[Listar os critérios para revisão dos produtos, enfatizando:

- *Fatores críticos para a qualidade do produto(s);*
- *Principais defeitos historicamente identificados no(s) produto(s);*
- *Fatores do(s) produto(s) que são base para realização de outras atividades.*

Criar uma seção e tabela de revisão para cada produto a ser revisado.]

3.1. VISÃO - <VERSÃO DO PRODUTO>

[A lista de itens de verificação descreve:

- **No:** Número seqüencial do item de revisão;
- **Descrição do Item:** Descrição do que deve ser revisado;
- **Situação:** situação do produto revisado em relação ao item a ser revisado:
 - **OK:** Produto(s) em conformidade com o item.
 - **NOK:** Produto(s) não conforme com a descrição do item. Ou seja, produto com defeito em relação ao item;
 - **NA:** Item não aplicável para o produto em revisão.
- **Observações e Defeitos:**
 - Quando situação **OK:** Campo deve permanecer em branco;
 - Quando situação **NOK:** Campo deve descrever o defeito encontrado, relatando inclusive a localização do defeito no produto (número de página, seção, etc);
 - Quando situação **NA:** Campo deve descrever o motivo pelo qual o item de revisão não se aplica ao produto sendo de revisado.
- **Gravidade:** Gravidade dos defeitos encontrados;
 - **Alta:** Defeitos que inviabilizem a compreensão do produto ou que caracterizam o produto como não aderente aos seus objetivos, como: Produto não atende aos requisitos, idéias impossíveis de serem realizadas, produto que cause impacto em outro produto já homologado, ausência de informações importantes no documento, etc.
 - **Média:** Defeitos que dificultam a compreensão do produto, porém este permanece adequado ao seu objetivo, como: erros de ortografia e gramática, poluição visual, mau encadeamento de idéias, etc.
 - **Baixa:** Defeitos relevantes que não afetam a compreensão ou adequação do produto aos seus objetivos, como: Detalhes de apresentação, cores, tamanhos de fontes, etc.

No	Descrição do Item	Situação			Observações e Defeitos	Gravidade
		OK	NOK	NA		
Geral						
1	O documento foi elaborado com base na última versão do modelo disponível?					
2	As informações do documento (nome, versão, autor, sigla e nome do projeto, etc) foram devidamente registradas nas propriedades deste?					
3	A versão do documento foi incrementada e a descrição da elaboração ou alteração foi registrada no histórico de revisões do documento?					
4	A capa do documento foi atualizada com o nome e versão do documento, sigla e nome do projeto e marca do cliente?					
5	O cabeçalho do documento foi atualizado com o nome do documento e marca do cliente?					
6	O rodapé do documento foi atualizado com a sigla e o nome do projeto?					
7	O sumário do documento foi atualizado?					
Requisitos						
8	O Problema está identificado e registrado de maneira clara?					
9	Os usuários do sistema estão identificados?					
10	As necessidades dos interessados estão identificadas? As colunas estão preenchidas corretamente? O nível de prioridade está definido para cada necessidade?					
11	As sentenças que nomeiam as Características estão descritas de acordo com o padrão indicado no documento Visão e no Processo de Gerenciamento e Engenharia de Requisitos?					
12	A descrição de cada Característica está clara? Basicamente, deve-se descrever como o recurso será percebido pelo usuário externamente, em alto nível. O público alvo deste texto são pessoas com formações heterogêneas e o objetivo é que todas entendam o sistema.					
13	Existem restrições impostas? Caso afirmativo, o entendimento está claro o suficiente?					
14	Os requisitos de documentação estão registrados?					

(1) Gravidade: Alta, Média, Baixa.

[Descrever todos defeitos de ortografia e gramática encontrados em qualquer seção do produto. Os defeitos de ortografia e gramática não são classificados quanto à gravidade, pois todos devem obrigatoriamente ser corrigidos.

- **Nº:** Número seqüencial do defeito
- **Localização do Defeito:** Número de página, parágrafo ou seção do produto na qual é localizado o defeito.
- **Descrição do Defeito:** Descrição do defeito de ortografia e gramática, sugere-se copiar o trecho de texto e destacar o defeito com cor de fonte diferente ou negrito.

]

Defeitos de Ortografia e Gramática		
No	Localização do Defeito	Descrição do Defeito

[Identificar sugestões de melhoria no produto revisado. Não há necessidade de implementação de todas as sugestões, ficando a cargo do Autor do produto decidir quanto à sua implementação.

- **Nº:** Número seqüencial da Sugestão
- **Descrição da Sugestão:** Descrição da sugestão de melhoria no produto revisado

]

Sugestões de Melhoria	
No	Descrição da Sugestão

APÊNDICE C – LISTA DE VERIFICAÇÃO II

PROCESSO DE DESENVOLVIMENTO DE SISTEMAS

EMBARCADOS

Logotipo da Empresa

Lista de Verificação de Especificação de Caso de Uso

<Sigla do Projeto> - <Nome do Projeto>		
Lista de Verificação de Especificação de Caso de Uso: <Nome do Documento>	Versão <X>	Data: 18/3/2008

1. IDENTIFICAÇÃO

[Descrever informações gerais sobre a revisão técnica sendo realizada:

- *Projeto: Sigla e nome do projeto ao qual o(s) produto(s) sendo revisados pertencem;*
- *Nome do Revisor: Nome da pessoa responsável pela revisão;*
- *Data da Revisão: Data da realização da revisão;*
- *Tempo de duração da revisão: Tempo gasto, em horas e minutos, com a atividade de revisão técnica do produto.]*

Projeto	<Sigla do Projeto> - <Nome do Projeto>		
Nome do Revisor			
Data da Revisão	<dd/MM/AAAA>	Tempo de Duração da Revisão	<X> horas e <X> minutos

2. OBJETIVO

[Descrever o objetivo da revisão e listar os produtos a serem revisados.]

Propósito da Revisão	
Revisar o Caso de Uso identificando desvios relativos ao detalhamento de requisitos funcionais e rastreabilidade.	
Produtos a serem revisados	Versão
<Nome da Especificação>	<X>

3. ITENS DE VERIFICAÇÃO

[Listar os critérios para revisão dos produtos, enfatizando:

- o Fatores críticos para a qualidade do produto(s);
- o Principais defeitos historicamente identificados no(s) produto(s);
- o Fatores do(s) produto(s) que são base para realização de outras atividades.

Criar uma seção e tabela de revisão para cada produto a ser revisado.]

3.1. CASO DE USO <NOME DO CASO DE USO> - <VERSÃO DO PRODUTO>

[A lista de itens de verificação descreve:

- **No:** Número seqüencial do item de revisão;
- **Descrição do Item:** Descrição do que deve ser revisado;
- **Situação:** situação do produto revisado em relação ao item a ser revisado:
 - o **OK:** Produto(s) em conformidade com o item.
 - o **NOK:** Produto(s) não conforme com a descrição do item. Ou seja, produto com defeito em relação ao item;
 - o **NA:** Item não aplicável para o produto em revisão.
- **Observações e Defeitos:**
 - o Quando situação **OK:** Campo deve permanecer em branco;
 - o Quando situação **NOK:** Campo deve descrever o defeito encontrado, relatando inclusive a localização do defeito no produto (número de página, seção, etc);
 - o Quando situação **NA:** Campo deve descrever o motivo pelo qual o item de revisão não se aplica ao produto sendo de revisado.
- **Gravidade:** Gravidade dos defeitos encontrados;
 - o **Alta:** Defeitos que inviabilizem a compreensão do produto ou que caracterizam o produto como não aderente aos seus objetivos, como: Produto não atende aos requisitos, idéias impossíveis de serem realizadas, produto que cause impacto em outro produto já homologado, ausência de informações importantes no documento, etc.
 - o **Média:** Defeitos que dificultam a compreensão do produto, porém este permanece adequado ao seu objetivo, como: erros de ortografia e gramática, poluição visual, mau encadeamento de idéias, etc.
 - o **Baixa:** Defeitos relevantes que não afetam a compreensão ou adequação do produto aos seus objetivos, como: Detalhes de apresentação, cores, tamanhos de fontes, etc.

No	Descrição do Item	Situação			Observações e Defeitos	Gravidade
		OK	NOK	NA		
Geral						

1	O documento foi elaborado com base na última versão do modelo disponível?				
2	As informações do documento (nome, versão, autor, sigla e nome do projeto, etc) foram devidamente registradas nas propriedades deste?				
3	A versão do documento foi incrementada e a descrição da elaboração ou alteração foi registrada no histórico de revisões do documento?				
4	A capa do documento foi atualizada com o nome e versão do documento, sigla e nome do projeto e marca do cliente?				
5	O cabeçalho do documento foi atualizado com o nome do documento e marca do cliente?				
6	O rodapé do documento foi atualizado com a sigla e o nome do projeto?				
Requisitos					
7	A descrição do DEC está clara e completa? Relata em poucas palavras o objetivo de negócio do Documento de Especificação de Caso de Uso e cita suas funcionalidades? Você entendeu o objetivo do DEC sem precisar ler os seus fluxos de eventos?				
8	Verifique se o Caso de Uso se encontra na lista ou Modelo de Casos de Uso existente no DRS.				
9	Procure termos vagos e peça esclarecimento (algum, às vezes, usualmente, freqüentemente).				
10	Todos os atores que interagem com o DEC foram relacionados?				
11	Verificar, para cada um dos passos do DEC, onde pode ocorrer uma validação de Regra de Negócio. Se ocorrer isso, incluir no final do passo a indicação para a Regra que está no documento de Regras de Negócio.				
12	Todos os atributos de entrada e de saída e regras relacionadas estão especificados? Exemplos: 1. Campos que deverão ser apresentados para o usuário. 2. Campos que deverão ser informados pelo usuário. 3. Regras de negócio que estabeleçam atributos de entrada ou saída.				
13	A obrigatoriedade dos atributos de entrada está especificada?				
14	Os atributos que serão selecionados a partir de uma lista de valores estão evidenciados? (Ex: O Ator seleciona a UF a partir de uma lista)				
15	Os valores das listas ou domínios específicos do sistema estão documentados no DEC?				
16	Conferir se o texto das mensagens referenciadas no DEC está apropriado para o negócio. Evitar mensagens longas demais ou mensagens curtas que não dizem nada.				
17	Se o DEC possui integrações, os requisitos destas (parâmetros, regras) estão documentados?				
Padrões e Formas					
18	O nome do DEC deve estar no infinitivo do verbo "Rejeitar...", "Receber..."				
19	Todos os documentos de referências foram listados no item referência. (Outros Casos de Uso, Glossário, Visão, etc.)				
20	Se algum dos itens do template do Documento de Especificação de Casos de Uso não tiver informação, esses itens não podem ser removidos. Informar para o item referente a seguinte frase: "Não se aplica".				

(1) Gravidade: Alta, Média, Baixa.

[Descrever todos defeitos de ortografia e gramática encontrados em qualquer seção do produto. Os defeitos de ortografia e gramática não são classificados quanto à gravidade, pois todos devem obrigatoriamente ser corrigidos.

- **Nº:** Número seqüencial do defeito
- **Localização do Defeito:** Número de página, parágrafo ou seção do produto na qual é localizado o defeito.
- **Descrição do Defeito:** Descrição do defeito de ortografia e gramática, sugere-se copiar o trecho de texto e destacar o defeito com cor de fonte diferente ou negrito.

]

Defeitos de Ortografia e Gramática		
No	Localização do Defeito	Descrição do Defeito

[Identificar sugestões de melhoria no produto revisado. Não há necessidade de implementação de todas as sugestões, ficando a cargo do Autor do produto decidir quanto à sua implementação.

- **Nº:** Número seqüencial da Sugestão
- **Descrição da Sugestão:** Descrição da sugestão de melhoria no produto revisado

]

Sugestões de Melhoria	
No	Descrição da Sugestão

APÊNDICE D – LISTA DE VERIFICAÇÃO III

PROCESSO DE DESENVOLVIMENTO DE SISTEMAS
EMBARCADOS

Logotipo da Empresa

Lista de Verificação de Especificação Suplementar

<Sigla do Projeto> - <Nome do Projeto>		
Lista de Verificação de Especificação Suplementar: <Nome do Documento>	Versão <X>	Data: 18/3/2008

1. IDENTIFICAÇÃO

[Descrever informações gerais sobre a revisão técnica sendo realizada:

- *Projeto: Sigla e nome do projeto ao qual o(s) produto(s) sendo revisados pertencem;*
- *Nome do Revisor: Nome da pessoa responsável pela revisão;*
- *Data da Revisão: Data da realização da revisão;*
- *Tempo de duração da revisão: Tempo gasto, em horas e minutos, com a atividade de revisão técnica do produto.]*

Projeto	<Sigla do Projeto> - <Nome do Projeto>		
Nome do Revisor			
Data da Revisão	<dd/MM/AAAA>	Tempo de Duração da Revisão	<X> horas e <X> minutos

2. OBJETIVO

[Descrever o objetivo da revisão e listar os produtos a serem revisados.]

Propósito da Revisão	
Revisar a Especificação de Requisitos não Funcionais identificando desvios relativos ao detalhamento do requisito e rastreabilidade.	
Produtos a serem revisados	Versão
<Nome da Especificação>	<X>

3. ITENS DE VERIFICAÇÃO

[Listar os critérios para revisão dos produtos, enfatizando:

- o Fatores críticos para a qualidade do produto(s);
- o Principais defeitos historicamente identificados no(s) produto(s);
- o Fatores do(s) produto(s) que são base para realização de outras atividades.

Criar uma seção e tabela de revisão para cada produto a ser revisado.]

3.1. REQUISITO NÃO FUNCIONAL <NOME DO REQUISITO NÃO FUNCIONAL> - <VERSÃO DO PRODUTO>

[A lista de itens de verificação descreve:

- **No:** Número seqüencial do item de revisão;
- **Descrição do Item:** Descrição do que deve ser revisado;
- **Situação:** situação do produto revisado em relação ao item a ser revisado:
 - o **OK:** Produto(s) em conformidade com o item.
 - o **NOK:** Produto(s) não conforme com a descrição do item. Ou seja, produto com defeito em relação ao item;
 - o **NA:** Item não aplicável para o produto em revisão.
- **Observações e Defeitos:**
 - o Quando situação **OK:** Campo deve permanecer em branco;
 - o Quando situação **NOK:** Campo deve descrever o defeito encontrado, relatando inclusive a localização do defeito no produto (número de página, seção, etc);
 - o Quando situação **NA:** Campo deve descrever o motivo pelo qual o item de revisão não se aplica ao produto sendo de revisado.
- **Gravidade:** Gravidade dos defeitos encontrados;
 - o **Alta:** Defeitos que inviabilizem a compreensão do produto ou que caracterizam o produto como não aderente aos seus objetivos, como: Produto não atende aos requisitos, idéias impossíveis de serem realizadas, produto que cause impacto em outro produto já homologado, ausência de informações importantes no documento, etc.
 - o **Média:** Defeitos que dificultam a compreensão do produto, porém este permanece adequado ao seu objetivo, como: erros de ortografia e gramática, poluição visual, mau encadeamento de idéias, etc.
 - o **Baixa:** Defeitos relevantes que não afetam a compreensão ou adequação do produto aos seus objetivos, como: Detalhes de apresentação, cores, tamanhos de fontes, etc.

No	Descrição do Item	Situação			Observações e Defeitos	Gravidade
		OK	NOK	NA		
Geral						

1	O documento foi elaborado com base na última versão do modelo disponível?				
2	As informações do documento (nome, versão, autor, sigla e nome do projeto, etc) foram devidamente registradas nas propriedades deste?				
3	A versão do documento foi incrementada e a descrição da elaboração ou alteração foi registrada no histórico de revisões do documento?				
4	A capa do documento foi atualizada com o nome e versão do documento, sigla e nome do projeto e marca do cliente?				
5	O cabeçalho do documento foi atualizado com o nome do documento e marca do cliente?				
6	O rodapé do documento foi atualizado com a sigla e o nome do projeto?				
Requisitos					
7	Existem requisitos de Usabilidade? Estão claramente especificados?				
8	Existem requisitos de Confiabilidade? Estão claramente especificados?				
9	Existem requisitos de Desempenho? Estão claramente especificados?				
10	Existem requisitos de Suportabilidade? Estão claramente especificados?				
11	Existem Restrições de Projeto? Estão claramente especificados?				
12	Existem outros Requisitos de Produto? Estão claramente especificados?				
13	Os requisitos suplementares foram detalhados por meio do preenchimento da tabela da Planguage?				

(1) Gravidade: Alta, Média, Baixa.

[Descrever todos defeitos de ortografia e gramática encontrados em qualquer seção do produto. Os defeitos de ortografia e gramática não são classificados quanto à gravidade, pois todos devem obrigatoriamente ser corrigidos.

- **Nº:** Número seqüencial do defeito
- **Localização do Defeito:** Número de página, parágrafo ou seção do produto na qual é localizado o defeito.
- **Descrição do Defeito:** Descrição do defeito de ortografia e gramática, sugere-se copiar o trecho de texto e destacar o defeito com cor de fonte diferente ou negrito.

]

Defeitos de Ortografia e Gramática		
No	Localização do Defeito	Descrição do Defeito

[Identificar sugestões de melhoria no produto revisado. Não há necessidade de implementação de todas as sugestões, ficando a cargo do Autor do produto decidir quanto à sua implementação.

- **Nº:** Número seqüencial da Sugestão

- **Descrição da Sugestão:** *Descrição da sugestão de melhoria no produto revisado*

]

Sugestões de Melhoria	
No	Descrição da Sugestão

APÊNDICE E – LISTA DE PRIORIZAÇÃO

Logotipo da Empresa

PROCESSO DE DESENVOLVIMENTO DE SISTEMAS
EMBARCADOS

Lista de Priorização

<Sigla do Projeto> - <Nome do Projeto>		
Lista de Priorização: <Nome do Documento>	Versão <X>	Data: 18/3/2008

Lista de Priorização: <Nome do Documento>

Histórico da Revisão

Data	Versão	Descrição	Autor
<dd/mm/aa>	<x.x>	<detalhes>	<nome>

Índice Analítico

- 1. Introdução 147
 - 1.1. Priorização das Necessidades do Cliente 147

Lista de Priorização: <Nome do Documento>

1. INTRODUÇÃO

[A introdução da lista de priorização fornece uma visão da priorização que deve ser seguida no desenvolvimento nas necessidades do sistema. Ela deve incluir as necessidades do cliente e o seu respectivo nível de priorização].

1.1. PRIORIZAÇÃO DAS NECESSIDADES DO CLIENTE

[Esta subseção fornece a priorização de cada uma das necessidades do cliente. Para cada necessidade identificar sua priorização na tabela a seguir]

Necessidade	Prioridade
<i>[Necessidade 1].</i>	<i>[Nível de Prioridade da Necessidade Listada, pode ser: 1: Prioridade Baixa 2: Prioridade Média 3: Prioridade Alta</i>
<i>[Necessidade 2].</i>	<i>[Nível de Prioridade da Necessidade Listada, pode ser: 1: Prioridade Baixa 2: Prioridade Média 3: Prioridade Alta</i>
<i>[Necessidade 3].</i>	<i>[Nível de Prioridade da Necessidade Listada, pode ser: 1: Prioridade Baixa 2: Prioridade Média 3: Prioridade Alta</i>

APÊNDICE F – DOCUMENTO DE REFERÊNCIA CRUZADA

PROCESSO DE DESENVOLVIMENTO DE SISTEMAS
EMBARCADOS

Logotipo da Empresa

Referência Cruzada

<Sigla do Projeto> - <Nome do Projeto>		
Lista de Priorização: <Nome do Documento>	Versão <X>	Data: 18/3/2008

Documento de Referência Cruzada: <Nome do Documento>

Histórico da Revisão

Data	Versão	Descrição	Autor
<dd/mmm/aa>	<x.x>	<detalhes>	<nome>

Índice Analítico

1. Introdução **Erro! Indicador não definido.**
2. Referência Cruzada **Erro! Indicador não definido.**
 - 2.1. Necessidades X Características **Erro! Indicador não definido.**
 - 2.2. Características X Requisitos Funcionais **Erro! Indicador não definido.**
 - 2.3. Requisitos Funcionais X Requisitos Funcionais **Erro! Indicador não definido.**
 - 2.4. Requisitos Funcionais X Requisitos Não – Funcionais **Erro! Indicador não definido.**
 - 2.5. Requisitos Não - Funcionais X Requisitos Não - Funcionais **Erro! Indicador não definido.**
 - 2.6. Outros X Outros **Erro! Indicador não definido.**
3. Glossário **Erro! Indicador não definido.**

Documento de Referência Cruzada: <Nome do Documento>

1. INTRODUÇÃO

[Descrever os objetivos do documento e outras informações relevantes para o entendimento do seu conteúdo.]

1.1. REFERÊNCIA CRUZADA

[Descrever tantas referências cruzadas quantas forem necessárias. As referências cruzadas sugeridas são de Necessidades X Características, Características X Requisitos Funcionais, Requisitos Funcionais X Requisitos Funcionais, Requisitos Funcionais X Requisitos Não-Funcionais e Requisitos Não – Funcionais X Requisitos Não - Funcionais].

1.1.1. NECESSIDADES X CARACTERÍSTICAS

Os relacionamentos permitidos são: <i>D – Dependência</i> <i>C – Colaboração</i> <i>P - Prejudica</i>	Características			
	<i>CRT001</i>	<i>CRT002</i>	<i>...</i>	<i>CRTNNN</i>
<i>Necessidades</i>				
<i>NCD001</i>				
<i>NCD002</i>				
<i>...</i>				
<i>NCDNNN</i>				

1.1.2. CARACTERÍSTICAS X REQUISITOS FUNCIONAIS

Os relacionamentos permitidos são: ▪ <i>D – Dependência</i> ▪ <i>C – Colaboração</i> ▪ <i>P - Prejudica</i>	Requisitos Funcionais			
	<i>RFC001</i>	<i>RFC002</i>	<i>...</i>	<i>RFCNNN</i>
<i>Características</i>				
<i>CRT001</i>				
<i>CRT002</i>				
<i>...</i>				
<i>CRTNNN</i>				

1.1.3. REQUISITOS FUNCIONAIS X REQUISITOS FUNCIONAIS

Os relacionamentos permitidos	Requisitos Funcionais

são: <i>D – Dependência</i> <i>C – Colaboração</i> <i>P - Prejudica</i>	<i>RFC001</i>	<i>RFC002</i>	<i>...</i>	<i>RFCNNN</i>
<i>Requisitos Funcionais</i>				
<i>RFC001</i>				
<i>RFC002</i>				
<i>...</i>				
<i>RFCNNN</i>				

1.1.4. REQUISITOS FUNCIONAIS X REQUISITOS NÃO – FUNCIONAIS

Os relacionamentos permitidos são: <i>D – Dependência</i> <i>C – Colaboração</i> <i>P - Prejudica</i>	<i>Requisitos Não-Funcionais</i>			
	<i>RNF001</i>	<i>RNF002</i>	<i>...</i>	<i>RNFNNN</i>
<i>Requisitos Funcionais</i>				
<i>RFC001</i>				
<i>RFC002</i>				
<i>...</i>				
<i>RFCNNN</i>				

1.1.5. REQUISITOS NÃO - FUNCIONAIS X REQUISITOS NÃO - FUNCIONAIS

Os relacionamentos permitidos são: <i>D – Dependência</i> <i>C – Colaboração</i> <i>P - Prejudica</i>	<i>Requisitos Não-Funcionais</i>			
	<i>RNF001</i>	<i>RNF002</i>	<i>...</i>	<i>RNFNNN</i>
<i>Requisitos Não-Funcionais</i>				
<i>RNF001</i>				
<i>RNF002</i>				
<i>...</i>				
<i>RNFNNN</i>				

1.1.6. OUTROS X OUTROS

Os relacionamentos permitidos são: <i>D – Dependência</i> <i>C – Colaboração</i> <i>P - Prejudica</i>	<i>Outros</i>			
	<i>OTR001</i>	<i>OTR002</i>	<i>...</i>	<i>OTRNNN</i>
<i>Outros</i>				

<i>OTR001</i>				
<i>OTR002</i>				
<i>...</i>				
<i>OTRNNN</i>				

1.2. GLOSSÁRIO

[Descrever os termos e siglas usados no documento.]

APÊNDICE G – DOCUMENTO DE ESPECIFICAÇÃO SUPLEMENTAR

PROCESSO DE DESENVOLVIMENTO DE SISTEMAS
EMBARCADOS

Logotipo da Empresa

**Documento de Especificação
Suplementar**

<Sigla do Projeto> - <Nome do Projeto>		
Documento de Especificação Suplementar: <Nome do Documento>	Versão <X>	Data: 18/3/2008

Documento de Especificação Suplementar: <Nome do Documento>

Histórico da Revisão

Data	Versão	Descrição	Autor
<dd/mmm/aa>	<x.x>	<detalhes>	<nome>

Índice Analítico

1. Introdução	154
1.1. Definições, Acrônimos e Abreviações	154
1.2. Referências	154
2. Funcionalidade	154
2.1. <Requisito Funcional Um>	154
3. Usabilidade	154
3.1. <Requisito de Usabilidade Um>	154
4. Confiabilidade	155
4.1. <Requisito de Confiabilidade Um>	155
5. Desempenho	155
5.1. <Requisito de Desempenho Um>	155
6. Suportabilidade	155

6.1. <Requisito de Suportabilidade Um>	155
7. Restrições de Projeto	155
7.1. <Restrição de Projeto Um>	156
8. Outros Requisitos do produto	156
8.1. Padrões Aplicáveis	156
8.2. Requisitos do Sistema	156
8.3. Requisitos Ambientais	156
9. Componentes Comprados	156
10. Interfaces	156
10.1. Interfaces de Usuário	156
10.2. Interfaces de Hardware	156
10.3. Interfaces de Software	156
10.4. Interfaces de Comunicações	157
11. Requisitos de Licenciamento	157

Documento de Especificação Suplementar: <Nome do Documento>

1. INTRODUÇÃO

[A introdução da Especificação Suplementar deve fornecer uma visão geral de todo o documento. Ela deve incluir a finalidade, o escopo, as definições, os acrônimos, as abreviações, as referências e a visão geral desta Especificação Suplementar.]

A Especificação Suplementar captura os requisitos de sistema que não são capturados imediatamente nos Casos de Uso do Modelo de Casos de Uso. Entre os requisitos estão incluídos:

- Requisitos legais e reguladores, incluindo padrões de aplicativo.*
- Atributos de qualidade do Sistema a ser criado, incluindo requisitos de usabilidade, confiabilidade, desempenho e suportabilidade.*
- Outros requisitos, como Sistemas operacionais e ambientes, requisitos de compatibilidade e restrições de design.*

]

1.1. DEFINIÇÕES, ACRÔNIMOS E ABREVIações

[Esta subseção deve fornecer as definições de todos os termos, acrônimos e abreviações necessárias à adequada interpretação da Especificação Suplementar. Essas informações podem ser fornecidas mediante referência ao Glossário do projeto.]

1.2. REFERÊNCIAS

[Esta subseção deve fornecer uma lista completa de todos os documentos mencionados em qualquer outra parte da Especificação Suplementar. Cada documento deve ser identificado por título, número do relatório (se aplicável), data e organização de publicação. Especifique as fontes a partir das quais as referências podem ser obtidas. Essas informações podem ser fornecidas por um anexo ou outro documento.]

1.3. FUNCIONALIDADE

[Esta seção descreve os requisitos funcionais do sistema que são expressos no estilo de linguagem natural. Normalmente, ela é organizada por recurso, mas métodos de organização alternativos como, por exemplo, organização por usuário ou organização por subsistema, também podem ser apropriados. Entre os requisitos funcionais podem estar incluídos conjuntos de recursos, capacidades e segurança.]

1.3.1. <REQUISITO FUNCIONAL UM>

[A descrição do requisito deve ser feita aqui.]

1.4. USABILIDADE

[Esta seção deve incluir todos os requisitos que afetam a usabilidade. Estes são alguns exemplos]:

- especifique o tempo de treinamento necessário para que usuários normais e usuários com conhecimentos avançados se tornem produtivos em operações específicas*
- especifique períodos de tempo mensuráveis para tarefas típicas ou*
- especifique requisitos que estejam em conformidade com os padrões comuns de usabilidade.*

]

1.4.1. <REQUISITO DE USABILIDADE UM>

[A descrição do requisito.]

1.5. CONFIABILIDADE

[Os requisitos de confiabilidade do sistema devem ser especificados aqui. Abaixo, algumas sugestões]:

- *Disponibilidade - especifique a porcentagem de tempo disponível (xx.xx%), as horas de uso, o acesso à manutenção, as operações de modo degradado etc.*
- *Tempo Médio entre Falhas (MTBF) - normalmente especificado em horas, mas também poderá ser especificado em termos de dias, meses ou anos.*
- *Tempo Médio para Reparo (MTTR) - quanto tempo o sistema poderá ficar sem funcionar após uma falha?*
- *Exatidão - especifique a precisão (resolução) e exatidão (através de algum padrão conhecido) necessárias na saída dos sistemas.*
- *Taxa máxima de erros ou defeitos - geralmente expressa em termos de erros/KLOC (thousands of lines of code, milhares de linhas de código) ou de erros/ponto de função.*
- *Taxa de erros ou defeitos - categorizados em termos de erros pouco importantes, importantes e críticos: o(s) requisito(s) deve definir o que se entende por um erro "crítico" (ex: perda total de dados ou total incapacidade de usar determinadas partes da funcionalidade do sistema).*

]

1.5.1. <REQUISITO DE CONFIABILIDADE UM>

[A descrição do requisito.]

1.6. DESEMPENHO

[As características de desempenho do sistema devem ser descritas nesta seção. Inclua tempos de resposta específicos. Quando aplicável, faça referência, por nome, aos Casos de Uso relacionados].

- *Tempo de resposta de uma transação (médio, máximo)*
- *Taxa de transferência (ex: transações por segundo)*
- *Capacidade (ex: o número de clientes ou de transações que podem ser acomodados pelo sistema)*
- *Modos de degradação (o modo aceitável de operação quando o sistema tiver sido degradado de alguma maneira)*
- *Utilização de recursos: memória, disco, comunicações etc.]*

1.6.1. <REQUISITO DE DESEMPENHO UM>

[A descrição do requisito.]

1.7. SUPORTABILIDADE

[Esta seção indica todos os requisitos que aprimorarão a suportabilidade ou manutenibilidade do sistema que está sendo criado, incluindo padrões de codificação, convenções de nomeação, bibliotecas de classes, acesso à manutenção e utilitários de manutenção.]

1.7.1. <REQUISITO DE SUPORTABILIDADE UM>

[A descrição do requisito.]

1.8. RESTRIÇÕES DE PROJETO

[Esta seção deve indicar todas as restrições de design referentes ao sistema que está sendo criado. As restrições de design representam decisões de design que foram impostas e devem ser obedecidas. Entre os exemplos desse tipo de restrição estão linguagens de software, requisitos de processo de software, uso prescrito de ferramentas de desenvolvimento, restrições de design e de arquitetura, componentes comprados, bibliotecas de classes etc.]

1.8.1. <RESTRIÇÃO DE PROJETO UM>

[A descrição do requisito deve ser feita aqui.]

1.9. OUTROS REQUISITOS DO PRODUTO

[Em um nível superior, liste padrões aplicáveis, requisitos de hardware ou de plataforma, requisitos de desempenho e requisitos ambientais.]

1.9.1. PADRÕES APLICÁVEIS

[Liste todos os padrões com os quais o produto deverá estar em conformidade. Entre eles, poderão estar incluídos padrões legais e reguladores, padrões de comunicações (TCP/IP, ISDN), padrões de conformidade com plataformas (Windows, UNIX etc) e padrões de qualidade e de segurança (ISO, CMMI).]

1.9.2. REQUISITOS DO SISTEMA

[Defina todos os requisitos do sistema necessários para suportar o aplicativo. Entre eles, poderão estar incluídos os sistemas operacionais de host e as plataformas de rede suportadas, configurações, memória, periféricos e software fornecido.]

1.9.3. REQUISITOS AMBIENTAIS

[Descreva os requisitos ambientais quando necessário. Para sistemas baseados em hardware, as questões ambientais poderão incluir temperatura, choques, umidade, radiação etc. Para aplicativos de software, os fatores ambientais podem incluir condições de uso, ambiente do usuário, disponibilidade de recursos, problemas de manutenção, e recuperação e tratamento de erros.]

2. COMPONENTES COMPRADOS

[Esta seção descreve todos os documentos comprados a serem usados no sistema, quaisquer restrições de utilização ou de licenciamento aplicáveis e quaisquer padrões associados de compatibilidade/interoperabilidade ou de interface.]

3. INTERFACES

[Esta seção define as interfaces que devem ser suportadas pelo aplicativo. Ela deve conter especificidades, protocolos, portas e endereços lógicos adequados, entre outros, para que o software possa ser desenvolvido e verificado em relação aos requisitos de interface.]

3.1. INTERFACES DE USUÁRIO

[Descreva as interfaces de usuário que deverão ser implementadas pelo software.]

3.2. INTERFACES DE HARDWARE

[Esta seção define todas as interfaces de hardware que devem ser suportadas pelo software, incluindo a estrutura lógica, os endereços físicos, o comportamento esperado, etc.]

3.3. INTERFACES DE SOFTWARE

[Esta seção descreve as interfaces de software com outros componentes do sistema de software. Poderão ser componentes comprados, componentes reutilizados de outro aplicativo ou componentes que estão sendo desenvolvidos para subsistemas fora do escopo desta SRS, mas com os quais esse aplicativo de software deve interagir.]

3.4. INTERFACES DE COMUNICAÇÕES

[Descreva todas as interfaces de comunicações com outros sistemas ou dispositivos como, por exemplo, redes locais, dispositivos seriais remotos etc.]

4. REQUISITOS DE LICENCIAMENTO

[Esta seção define todos os requisitos de imposição de licenciamento ou outros requisitos de restrição de utilização que devem ser exibidos pelo software.]

5. DETALHAMENTO DOS REQUISITOS NÃO-FUNCIONAIS

[Esta seção define os atributos de qualidade de todos os requisitos não-funcionais. O objetivo da seção é gerar uma especificação completa de todos os requisitos não-funcionais do sistema. Cada um dos requisitos não-funcionais identificados deve ser especificado em uma tabela.]

5.1. <REQUISITO NÃO-FUNCIONAL UM>

Palavras-Chave	Descrição
Tipo	<i>[Etiqueta, rótulo, identificador persistente e único do requisito].</i>
Descrição	<i>[Descrição simples e breve do conceito principal ou significado geral do requisito].</i>
Stakeholder	<i>[Envolvidos/Afetados pelo requisito].</i>
Escala	<i>[Escala usada para quantificar o requisito].</i>
Métrica	<i>[Processo ou método para medir escalas dos requisitos].</i>
Método	<i>[Método para medir a escala].</i>
Frequência	<i>[Frequência para medição].</i>
Responsável	<i>[Pessoas/Departamento responsável por fazer as medições].</i>
Registro	<i>[Onde/Quando as medidas devem ser reportadas].</i>
Nível Mínimo	<i>[O nível mínimo requerido para evitar falhas].</i>
Plano	<i>[Nível para obter sucesso exigido].</i>
Nível Sucesso	<i>[Como prolongar, aumentar, alongar o sucesso].</i>
Nível Desejado	<i>[Nível desejável de sucesso que não pode ser atingido através dos métodos atuais].</i>
Histórico	<i>[Resultados anteriores para comparação (histórico)].</i>
Tendência	<i>[Tendência histórica].</i>
Histórico de Sucesso	<i>[O melhor resultado obtido].</i>
Definição	<i>[Definição oficial do termo].</i>
Autoridade	<i>[Pessoa, grupo ou nível de autorização].</i>