AVALIAÇÃO DA QUALIDADE DE SISTEMAS ESPECIALISTAS

Káthia Marçal de Oliveira

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profª Ana Regina Cavalcanti da Rocha, D.Sc.
(Presidente)

Prof. Gerson Zaverucha, PhD.

Prof. Álvaro Rabelo Alves Junior, L.D.

Prof. Crediné Silva de Menezes, D.Sc.

Rio de Janeiro, RJ - Brasil
Março de 1995

Oliveira, Káthia Marçal de

Avaliação da Qualidade de Sistemas Especialistas (Rio de Janeiro 1995)
170 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1995)
Tese - Universidade Federal do Rio de Janeiro, COPPE
1. Avaliação da Qualidade de Software
I. COPPE/UFRJ II. Título (série)

A meus pais e
meus irmãos

AGRADECIMENTOS

À Ana, por realizar mais que um maravilhoso trabalho de orientação, por ser a chefe exigente, a amiga carinhosa, a conselheira paciente e uma das principais responsáveis por tudo que sou.

Ao Dr. Álvaro, pela confiança depositada em mim ao me oferecer o ambiente para realizar minha tese, pela valiosa colaboração ao meu trabalho e por me ter ensinado que a paixão pelo que se faz é a garantia do sucesso.

À Vera, que me mostrou que o melhor trabalho é aquele realizado não apenas em conjunto mas em cooperação mútua, que foi a colega, a professora, a amiga, a defensora, e que fez dos momentos em que passamos juntas um constante aprendizado de vida.

Aos meus pais que com o amor que sentem por mim, mesmo sem entender os meus propósitos, aceitaram e apoiaram minhas decisões.

Ao Val, por ser mais que meu irmão, ser meu melhor amigo, meu conselheiro, meu cúmplice.

Ao Júnior, que com sua tranquilidade, suportou os meus medos e minhas crises e me ofereceu seu carinho, sua atenção e, principalmente, seu amor.

À Gê, que acompanhou todo o meu trabalho, compartilhando segredos, inseguranças, alegrias, vitórias e tudo aquilo que só os verdadeiros amigos compartilham.

Ao George, que antes mesmo de mim, acreditou que este era o caminho que iria me tornar feliz, e mesmo à distância sempre me apoiou e comemorou comigo minhas conquistas.

Ao Jorge, que me fez sentir em família, me ofereceu sua casa, sua amizade e o delicioso feijãozinho carioca.

À Lu, minha amiga de todas as horas, a Carol e Xexéo por me mostrarem o lado alegre e festivo da cidade maravilhosa.

À Hermê, pelo carinho e as comidinhas sempre gostosas.

À Neide, pelas conversas sempre sensatas.

Aos amigos que conquistei, especialmente, a Blasheck, Renata, Guilherme, Gilda, Clifton, Francisco, Belchior, Raquel e Washington.

À Cláudia Werner pelas caronas e amizade.

Aos Jano, Gerson e Mário por contribuirem significativamente na minha formação.

Ao Prof. Crediné pela gentileza em aceitar participar da banca examinadora desta tese.

A Dr. Agnaldo, Dr. Ximenes, Dr. Nelson, Werther, Luiz Agnaldo, Mário e Glorinha que suportaram as minhas cobranças, foram pacientes e amigos levando o nosso trabalho ao resultado tão almejado e a Dulcinea que mesmo participando apenas no final, soube ver o valor do meu trabalho.

Ao Amorim, pelo convite para participar do convênio UFBA/UFRJ dando-me a oportunidade de realizar este trabalho.

Ao Maurício Cardeal, pela colaboração na análise estatística utilizada neste trabalho.

Aos funcionários do Programa pelo apoio imprescindível à realização do trabalho de todos os alunos.

À IBM-Brasil e ao CNPq pelo apoio finaceiro à realização deste trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M.Sc.)

AVALIAÇÃO DA QUALIDADE DE SISTEMAS ESPECIALISTAS

Káthia Marçal de Oliveira

Março, 1995

Orientadora: Ana Regina Cavalcanti da Rocha, D. Sc.

Programa: Engenharia de Sistemas e Computação

Uma razão para a limitação de uso de sistemas especialistas pode ser atribuída à sua qualidade, muitas vezes insuficiente.

A qualidade de um sistema especialista está relacionada à qualidade do modelo que representa o raciocínio humano e a um conjunto de atributos relacionados à sua qualidade externa (percebida por seus usuários) e à sua qualidade interna (percebida por seus desenvolvedores e mantenedores). Avaliar a qualidade de um sistema especialista, quanto ao modelo de raciocínio e a este conjunto de atributos não é uma questão resolvida.

Esta tese define um conjunto de atributos de qualidade a serem considerados na avaliação de um sistema especialista, organizando-os segundo o modelo proposto por Rocha.

Finalmente este conjunto de atributos unido a técnicas para verificação e validação de sistemas especialistas foi utilizado no planejamento e avaliação da qualidade de um sistema especialista para diagnóstico em Cardiologia ao longo de seu processo de desenvolvimento. Os resultados obtidos com esta experiência são, também, relatados na tese.

Abstract of thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for degree of Master Science (M.Sc.)

SOFTWARE QUALITY EVALUATION FOR EXPERT SYSTEMS

Káthia Marçal de Oliveira

March, 1995

Thesis supervisor: Ana Regina Cavalcanti da Rocha, D. Sc.

Department: Systems and Computer Engineering

A major reason for the limited use of expert systems is their poor quality.

The quality of an expert system depends on the quality of the model which represents human reasoning and on a set of attributes related to its external quality (as perceived by its users) and to its internal quality (as perceived by the development and maintenance team). Questions concerning the evaluation of the quality of an expert system by the rational thinking model and by a set of attributes have not been fully resolved.

This thesis defines a set of quality attributes to be considered in the evaluation of an expert system according to a model proposed by Rocha.

Finally these attributes and several techniques for expert systems verification and validation were used for quality planning and evaluation of an expert system in cardiology, during its development process. The results obtained with this experience are also described in this thesis.

SUMÁRIO

Lista de Figuras

Capítulo 1
Introdução

Com a consolidação da Informática e a importância de sua utilização em diferentes áreas, aumentou a busca por melhores métodos, técnicas e ferramentas que apoiassem o desenvolvimento de software. Neste sentido, a garantia da qualidade no processo de desenvolvimento surgiu como uma necessidade essencial para a produção de software de qualidade. Entretanto, para termos software de alta qualidade não basta preocupar-nos com a qualidade à nível do processo. Qualidade se atinge através do uso de um processo de desenvolvimento adequado e de procedimentos de garantia da qualidade a nível do próprio produto [Bazzana 93].

Sistemas especialistas são programas responsáveis pela solução de problemas de domínios especializados nos quais a solução efetiva do problema geralmente requer um especialista humano. As diferenças entre software convencional e sistemas especialistas determinam a necessidade de peculiaridades no processo de verificação e validação destes sistemas [Messeguer 93].

A utilização de um processo adequado de garantia de qualidade, certamente conduzirá a sistemas especialistas com menos riscos de falhas. Entretanto, nota-se a carência de métodos e técnicas para avaliação da qualidade adequados a estes sistemas.

Para se conseguir progresso em direção a uma efetiva metodologia para avaliação de sistemas especialistas é necessária a realização de experimentos com técnicas adequadas à sua avaliação. Incentivados à experimentação, através da participação no projeto de desenvolvimento de um sistema especialista para Cardiologia, definimos um conjunto de atributos de qualidade para avaliação de sistemas especialistas, considerando a estrutura do modelo proposto por Rocha [Rocha 83]. O conjunto de atributos definidos foi identificado a partir de trabalhos anteriores realizados na COPPE, da norma ISO 9126 e de uma revisão da literatura específica.

Este trabalho insere-se, portanto, no contexto do projeto *Sistemas Especialistas em Cardiologia* da Unidade de Cardiologia e Cirurgia Cardiovascular do Hospital Universitário Prof. Edgard Santos da Universidade Federal da Bahia/Fundação Bahiana de Cardiologia (Projeto FINEP nº 66940058-00). Assim sendo, a tese foi desenvolvida

na UCCV/FBC, através de convênio entre a mesma e a área de Engenharia de Software da COPPE.

Esta tese está organizada em cinco capítulos, além desta introdução.

No capítulo 2 ressaltamos a importância e a evolução do estado da arte em qualidade de software, discutindo aspectos referentes à qualidade e ao processo de garantia desta qualidade. É discutida a importância do planejamento e da gerência da qualidade. Finalmente, são discutidas técnicas a serem utilizadas no processo de garantia da qualidade.

No capítulo 3, é explorado o estado da arte com relação a conceitos, métodos e técnicas relacionados, especificamente, à questão da avaliação da qualidade de sistemas especialistas.

No capítulo 4, é definido um conjunto de atributos de qualidade a serem considerados na avaliação de um sistema especialista, tanto no que se refere ao produto final, quanto aos diferentes produtos gerados durante o desenvolvimento.

No capítulo 5, é descrita uma experiência de planejamento e avaliação da qualidade de um sistema especialista para diagnóstico médico, onde foram utilizados os conceitos discutidos nos capítulos 2 e 3 e os atributos de qualidade definidos no capítulo 4.

Finalmente, no capítulo 6, apresentamos as conclusões deste trabalho.

Capítulo 2
Qualidade de Software

2.1 Introdução

Com o crescimento do volume e complexidade das aplicações de software, também tem se evidenciado a importância da qualidade do mesmo. A questão da baixa qualidade tem sido um discutido tema no desenvolvimento de software, sendo colocado no ápice da problemática relacionada à crise do sofware. Os progressos nesta área tem sido significativos, mas não o suficiente. Entretanto, já se pode notar nas organizações um esforço no sentido de adotarem princípios de garantia da qualidade capazes de assegurar a construção de produtos de melhor qualidade.

Neste capítulo, são discutidos aspectos relacionados à qualidade de software e ao processo de garantia desta qualidade.

2.2 Contexto Histórico e Tendências

A construção de software exige, cada vez mais, um processo sistemático e controlado. O mercado consumidor tem se tornado mais rigoroso na escolha de seus produtos e a qualidade dos mesmos passou a ser um fator crítico para a garantia do sucesso. Por outro lado, desenvolvedores de software sentem a necessidade de melhorar a qualidade de seus produtos para manterem-se competitivos no mercado.

A própria evolução histórica do desenvolvimento de software já previa uma época em que a simples existência de software não satisfaria o mercado, que se tornaria mais exigente no sentido de não apenas produzir ou comprar produtos de software, e sim de produzir e comprar software de qualidade [Basili 91].

A década de 60 ficou conhecida como a era funcional pela introdução da tecnologia da informação nas instituições e pelo início da adequação de seus serviços e funções com os softwares. Nos anos 70, a introdução de modelos de ciclo de vida buscou suprir a

necessidade do mercado com relação ao desenvolvimento de software de forma planejada, controlada e que atendesse a prazos estabelecidos. Na década subsequente, a queda progressiva do preço do hardware impulsionou a preocupação com a busca de um menor custo de desenvolvimento de sofware. O uso de modelos para estimativa de custos e recursos expandiu-se e a importância da produtividade no desenvolvimento de software cresceu substancialmente [Basili 91].

No momento atual, a qualidade de software é trazida para o centro do processo de desenvolvimento. Com o maior amadurecimento dos softwares do mercado, maior é a exigência de garantia da qualidade por parte dos usuários. A tendência das instituições é reduzir o número de seus fornecedores de software, excluindo dos seus negócios aqueles que não estão aptos a fornecer qualidade em seus produtos. Algumas instituições, como por exemplo o governo e multinacionais da Europa, já utilizam como pré-requisito para escolha de seus fornecedores de software, a certificação segundo o padrão de qualidade da ISO [Miller 93], [Sanders 94].

Collins [Collins 94] considera que o processo de desenvolvimento de software, desde sua criação até seu uso efetivo é um processo social que envolve compradores, desenvolvedores, usuários e outras pessoas que possam ser afetadas pela utilização do software que ele denomina de *penumbra*. Nesse contexto, propõe que cada uma dessas pessoas tenham responsabilidades éticas para minimizar os riscos encontrados e derivados de suas participações nesse processo, e com isso contribuir para melhorar a qualidade do produto. Essas responsabilidades são traduzidas a partir da definição de princípios que definem que o software não deve aumentar o grau de risco (financeiro ou pessoal), em situações já vulneráveis por causa da falta de conhecimento sobre o software.

Padrões de qualidade estão sendo estabelecidos garantindo aos compradores o estabelecimento de exigências de desempenho do software determinados em contrato, que podem levar os desenvolvedores a receberem acusações de negligência, desvinculação de propósitos e mau profissionalismo no caso do não atendimento das exigências, tendo portanto, que estarem conscientes de padrões legais de competição quando construírem seus produtos. Segundo Collins [Collins 94], considerações éticas no desenvolvimento de software podem não só unificar aspectos destes padrões legais, como contribuir de forma significativa para melhorar e padronizar a questão da qualidade de software.

Um fator que contribui para a expansão da política de qualidade é o reconhecimento de que desenvolvedores de software que possuem um sistema efetivo de garantia de

qualidade têm conseguido um aumento da produtividade e redução de custos no desenvolvimento de seus produtos [Ackerman 89].

É certo que o custo da correção tardia de defeitos de software é muito maior do que se esta correção for realizada no início do processo de desenvolvimento do software [Sanders 94], [Pressman 92], [Möller 93]. A redução de custos de desenvolvimento é, portanto, alcançada pelo redirecionamento de recursos para a prevenção de erros. A introdução de um sistema de garantia da qualidade é um passo essencial nesse sentido. Além disso, o custo inicial para estabelecimento de um sistema de qualidade é logo compensado com o resultado gerado no trabalho.

A necessidade de qualidade em software está, portanto, conduzindo o mercado atual tanto internamente nas empresas no desenvolvimento de seus produtos, quanto externamente no ambiente competitivo de negócios. Deming [in Sanders 94] sugere que esse processo é uma *reação em cadeia* como mostra a Figura 2.1: melhorar a qualidade no desenvolvimento de software implica em melhorar a produtividade o que leva a redução de custos, com isso, aumenta-se o mercado, havendo um crescimento nos negócios da empresa o que gera o retorno dos investimentos.

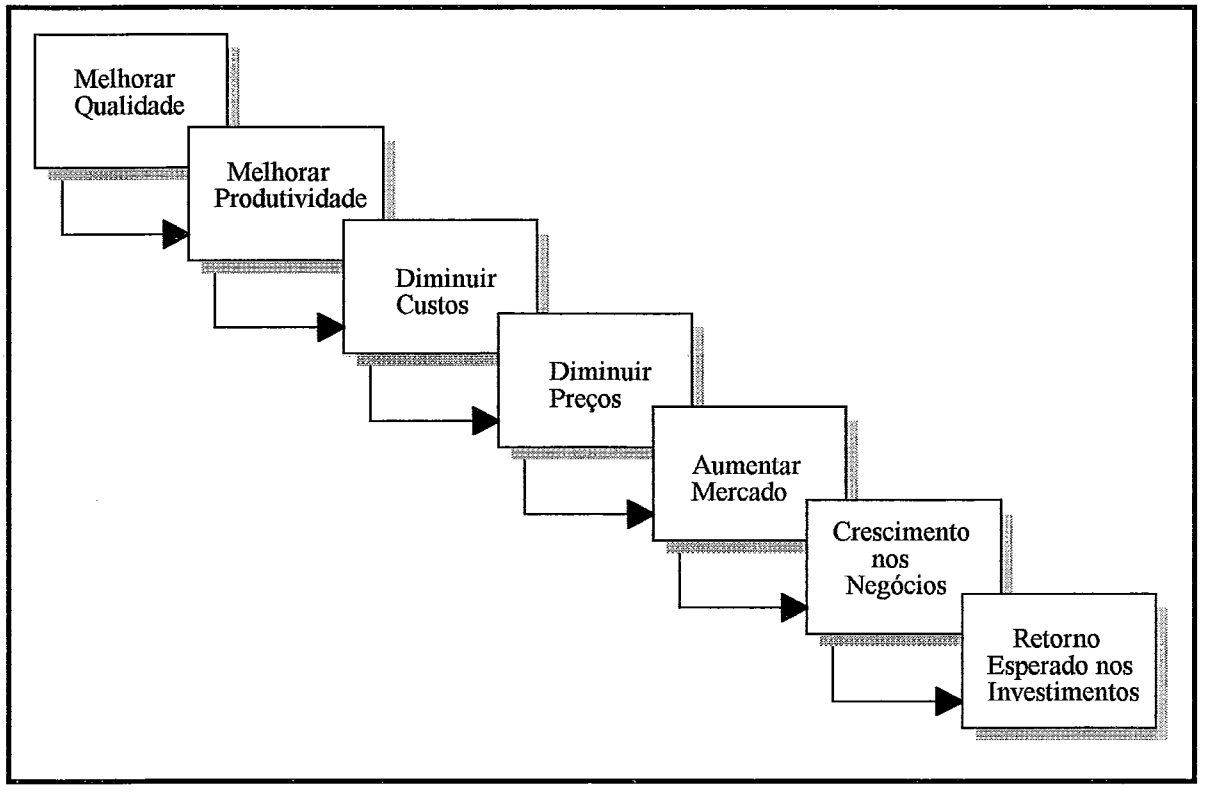

Figura 2.1 - Reação em cadeia de Deming [Sanders 94]

2.2.1 A Questão da Qualidade de Software no Brasil

Com o crescimento do mercado internacional e a exigência de melhores produtos, sente-se a necessidade de inserir o Brasil neste contexto através de uma profunda modernização industrial e na infra-estrutura governamental, sendo indispensável qualidade e produtividade. Devido a esta preocupação, o Governo propôs à sociedade, o Programa Brasileiro de Qualidade e Produtividade (PBQP) que tem como objetivo estabelecer um conjunto ordenado de ações indutoras da modernização industrial e tecnológica, contribuindo para a retomada do desenvolvimento econômico e social [Weber 94a]. Este programa é composto de subprogramas e projetos de abrangência geral e setorial, sendo executados de forma descentralizada pelos vários agentes econômicos e com uma atuação empresarial voltada para a qualidade e produtividade.

Recentes pesquisas, no Brasil, evidenciaram que a maior parte das empresas ainda não apresentam maturidade no tratamento da qualidade em software [Weber 94a]. No entanto, com essas pesquisas, realizadas em 1990 e 1993, pôde-se constatar que a preocupação com a qualidade no desenvolvimento de software está em um processo de evolução. Aos poucos as empresas brasileiras estão procurando pesquisar e adotar práticas de garantia da qualidade no desenvolvimento dos seus projetos devido a um clima favorável para melhoria de qualidade e da produtividade e a evidência do sucesso dessas práticas em algumas empresas.

Em 1990, constatou-se que poucas empresas adotavam métodos de gestão voltados para a qualidade e uma grande proporção não considerava sequer a possibilidade de vir a adotar procedimentos formais para medir e assegurar a qualidade do software desenvolvido. Além disso, o número de pessoas qualificadas para o controle da qualidade no desenvolvimento de softwares era insignificante.

Com a pesquisa realizada em 1993, observou-se que as empresas ainda não tem uma cultura voltada para gestão de qualidade (cerca de 50% afirmaram não ter um sistema de qualidade no desenvolvimento), que poucas medem de forma sistemática a qualidade de seus produtos ou serviços (apenas 26%), que há pouca adoção de plano de testes e dos resultados de revisões ou testes e que os procedimentos informais ou *ad-hoc* no desenvolvimento de software ainda são maioria. Entretanto, percebe-se que há uma evolução na preocupação com esses aspectos considerando-se a situação evidenciada em 1990. Constatou-se que uma parte das empresas já tem implantado programas de garantia de qualidade, e que uma grande parte está em processo de implantação ou estudando sua viabilidade. Da mesma forma, aumentou-se, no mercado, o número de profissionais capacitados para a implantação de programas de qualidade.

Com a crescente capacitação técnica dos profissionais, o reconhecimento da necessidade de se dispor de boa infra-estrutura de serviços tecnológicos e com isso usar métodos, ferramentas e procedimentos de Engenharia de Software no processo de desenvolvimento, a tendência do setor de produção de software é continuar evoluindo no que se refere à garantia da qualidade.

A tendência evidenciada na pesquisa é de aumentar o número de empresas que incluem, de forma sistemática, metas ou diretrizes para a qualidade nos seus planos estratégicos, um passo fundamental para a implantação efetiva de programas de garantia da qualidade.

Para atender a esta demanda de qualidade, muitos trabalhos estão sendo desenvolvidos em diferentes áreas de aplicação. Alguns destes trabalhos foram apresentados recentemente (1994) no VIII Simpósio Brasileiro de Engenharia de Software abordando qualidade de produto e qualidade do processo de desenvolvimento. Em relação a qualidade do produto foram discutidos aspectos que abrangem desde qualidade em áreas específicas (como sistemas especialistas [Oliveira 94a] e modelos orientados a objetos [Clunie 94]) até uma preocupação com o apoio ao engenheiro de software no que diz respeito a definição de requisitos [Fiorini 94], definição de atributos de qualidade de especificações [Belchior 94] e ao planejamento da qualidade em projetos [Passos 94]. A qualidade no processo de desenvolvimento foi enfatizada como essencial [Rabelo 94], [Rocha 94] destacando-se, ainda, a importância de modelos de gestão [Weber 94b].

2.3 Características de Qualidade de Software

Atualmente, é inquestionável que a qualidade é uma preocupação fundamental no desenvolvimento de software. Sanders [Sanders 94] valoriza e justifica essa preocupação afirmando que qualidade é crítica para a sobrevivência e o sucesso das empresas. Diante desse contexto, empresas que desenvolvem software estão buscando adotar procedimentos cuidadosos e rigorosos ao longo do processo de desenvolvimento de seus produtos para atingir a meta de qualidade em seus produtos.

Pelo vocabulário da ISO [ISO/CD 8402], qualidade é a totalidade das características de um produto ou serviço que lhe confere a capacidade de satisfazer às necessidades implícitas e explícitas de seus usuários. Essa definição, portanto, identifica qualidade com satisfação do usuário. Rocha [Rocha 87] sintetiza esse conceito definindo qualidade

de software como "um conjunto de propriedades a serem satisfeitas em determinado grau, de modo que o software satisfaça às necessidades de seus usuários".

Segundo Basili [Basili 91] qualidade é um conceito multidimensional cujas dimensões incluem a entidade de interesse (por exemplo: especificação de requisitos, projeto, produto final, etc), o ponto de vista sobre esta entidade (por exemplo: visão do desenvolvedor, visão do usário final, visão do gerentes, etc) e as caracteríticas de qualidade pertinentes a esta entidade.

O processo de garantia da qualidade de software está relacionado à qualidade de seu processo de desenvolvimento e às características de qualidade do próprio produto. Pesquisa recente [Bazzana 93] comprova o reconhecimento desta realidade por parte de desenvolvedores e usuários.

Procedimentos para garantia da qualidade de software em relação ao processo de desenvolvimento estão descritos na norma ISO 9000-3 [ISO 9000-3]. Esta norma consiste de três partes:

- *estrutura*, que descreve os aspectos organizacionais a serem considerados na produção de software (responsabilidade da administração, sistemas da qualidade, auditorias internas do sistema da qualidade, ação corretiva);
- *atividades do ciclo de vida*, que define as ações necessárias para conduzir cada fase do processo de desenvolvimento (revisão de contrato, especificação dos requisitos do cliente, planejamento do desenvolvimento, planejamento da qualidade, projeto e implementação, testes e validação, aceitação, cópia, entrega e manutenção);
- *atividades de apoio*, que define as atividades de suporte à produção, entrega e manutenção de software (gerenciamento de configuração, controle de documentos, registros da qualidade, medições, regras práticas e convenções, ferramentas e técnicas, compras, aquisição, produto de software incluído e treinamento).

Estes procedimentos não serão abordados neste trabalho pois fogem ao seu escopo. Uma análise desta norma e sua aplicação a sistemas especialistas pode ser encontrada em [Werneck 94], [Werneck 95].

As características de qualidade de produtos de software estão descritas na norma ISO 9126 [ISO/IEC 9126] que detalharemos a seguir.

2.3.1 Características de Qualidade de Software segundo a ISO 9126

Características de qualidade [ISO/CD 8402] ou propriedades [Rocha 87] são os atributos de um produto de software através das quais sua qualidade pode ser descrita e avaliada. Estas características devem estar relacionadas aos requisitos de qualidade estabelecidos para o produto.

Requisitos de qualidade para produtos de software, entretanto, são difíceis de definir. Vários fatores contribuem para esta dificuldade. Áreas de aplicação diferentes tem requisitos de qualidade diferentes. Além disso, diferentes projetos numa mesma área de aplicação podem apresentar requisitos de qualidade diferentes. Os requisitos de qualidade para uma aplicação podem, ainda, conflitar pois atingir um requisito em maior grau pode significar uma perda em outro requisito.

A norma ISO 9126 [ISO/IEC 9126] tem como objetivo definir as características de qualidade desejáveis para um produto de software. Define seis características genéricas de qualidade:

- *funcionalidade*, que refere-se à existência de um conjuntos de funções que satisfazem necessidades estabelecidas ou implícitas e suas propriedades específicas;
- *confiabilidade*, que refere-se à capacidade do software de manter seu nível de desempenho, sob condições estabelecidas, por um período de tempo estabelecido;
- *utilizabilidade*, que refere-se ao esforço necessário para se poder utilizar o software, bem como o julgamento individual desse uso, por um conjunto de usuários estabelecido ou subentendido;
- *eficiência*, que refere-se ao relacionamento entre o nível de desempenho do software e a quantidade de recursos utilizada, sob condições estabelecidas;
- *manutenibilidade*, que refere-se ao esforço necessário para fazer modificações específicas no software;
- *portatilidade*, que refere-se à habilidade do software ser transferido de um ambiente para outro.

Para cada uma destas características são identificadas e definidas sub-características (Figura 2.2).

Característica	Sub-característica	Descrição
Funcionalidade	Adequação	Atributos do software que evidenciam a presença de um conjunto de funções e sua adequação para as tarefas especificadas
	Acurácia	Atributos do software que evidenciam a geração de resultados ou efeitos corretos ou conforme esperados
	Interoperabilidade	Atributos do produto de software que evidenciam sua capacidade de interagir com sistemas especificados
	Conformidade	Atributos do software que fazem com que ele esteja de acordo com as normas, convenções ou regulamentações previstas em leis e descrições similares, relacionadas à aplicação
	Segurança	Atributos do software que evidenciam sua capacidade de evitar acesso não autorizado, acidental ou deliberado, a programas e dados.
Confiabilidade	Maturidade	Atributos do software que evidenciam a frequência de falhas por defeitos de software.
	Tolerância a falhas	Atributos do software que evidenciam sua capacidade em manter um nível de desempenho especificado no caso das falhas no software ou violação nas interfaces especificadas
	Recuperabilidade	Atributos do software que evidenciam sua capacidade de restabelecer seu nível de desempenho e recuperar os dados diretamente afetados, em caso de falha, e o tempo e esforço necessários para tal
Eficiência	Comportamento em relação ao tempo	Atributos do software que evidenciam seu tempo de resposta, tempo de processamento e velocidade na execução de suas funções
	Comportamento em relação aos recursos	Atributos do software que evidenciam a quantidade de recursos usados e a duração de seu uso na execução de suas funções

Figura 2.2 - Características e sub-características de qualidade de software da ISO 9126

Característica	Sub-característica	Descrição
Utilizabilidade	Inteligibilidade	Atributos do software que evidenciam o esforço do usuário para reconhecer o conceito lógico e sua aplicabilidade
	Apreensibilidade	Atributos do software que evidenciam o esforço do usuário para aprender sua aplicação (por exemplo: controle de operação, entradas, saídas)
	Operacionalidade	Atributos do software que evidenciam o esforço do usuário para sua operação e controle da sua operação
Manutenibilidade	Analisabilidade	Atributos do software que evidenciam o esforço necessário para diagnosticar deficiências ou causas de falhas, ou para identificar partes a serem modificadas
	Modificabilidade	Atributos do software que evidenciam o esforço necessário para modificá-lo, remover seus defeitos ou adaptá-lo a mudanças ambientais
	Estabilidade	Atributos do software que evidenciam o risco de efeitos inesperados ocasionados por modificações
	Testabilidade	Atributos do software que evidenciam o esforço necessário para validar o software modificado

Figura 2.2 (cont.) - Características e sub-características de qualidade de software da ISO 9126

Característica	Sub-característica	Descrição
Portatilidade	Adaptabilidade	Atributos de software que evidenciam sua capacidade de ser adaptado a diferentes ambientes especificados, sem a necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado
	Capacidade para ser instalado	Atributos de software que evidenciam o esforço necessário para sua instalação num ambiente especificado
	Conformidade	Atributos do software que o tornam consonante com padrões relacionados à portatilidade
	Capacidade para Substituir	Atributos do software que evidenciam sua capacidade e esforço necessário para substituir um outro software, no ambiente estabelecido para esse outro software.

Figura 2.2 (cont.) - Características e sub-características de qualidade de software da ISO 9126

Uma pesquisa realizada em vários países da Comunidade Européia [Bazzana 93], comprovou a importância já alcançada pela ISO 9126 (cerca de 70% dos entrevistados pelo menos já tinham ouvido falar da norma) embora esta seja uma norma muito recente. Bazzana conclui, portanto, que a ISO 9126 já é realmente *o padrão* para modelagem da qualidade de software.

No entanto, para se viabilizar a avaliação de software, segundo as características e sub-características identificadas na ISO 9126, é necessário correlacioná-las a critérios e métricas específicas, de preferência especializadas para os diversos domínios de aplicação.

2.3.2 Modelos para Avaliação da Qualidade

Qualidade de software é, portanto, alcançada através de um conjunto de características. Para que seja possível realizar avaliações em produtos de software é necessário organizar estas características de qualidade em modelos.

Existem na literatura vários modelos para a avaliação da qualidade de software: a Ciência de Software proposta por Maurice Halstead [Halstead 77] baseada na contagem de operadores e operandos que aparecem na implementação; o modelo proposto por Barry Boehm [Boehm 78] que define uma árvore de características de qualidade; e os modelos fruto das experiências da Computer Services Association [Möller 93] e do projeto SCOPE (Software Certification Programme in Europe) [Boegh 93].

No contexto deste trabalho consideramos para definição de características de qualidade e procedimentos para avaliação de software, o modelo proposto por Rocha [Rocha 83] que descrevemos a seguir. A escolha deste modelo deve-se ao fato desta tesa inserir-se, também, no Projeto Integrado do CNPq "Controle da Qualidade de Software" que tem como objetivo definir atributos de qualidade para diferentes domínios de aplicação usando a estrutura deste modelo.

2.3.2.1 Modelo proposto por Rocha

O modelo definido por Rocha foi influenciado por trabalhos anteriores de Boehm [Boehm 78] e McCall [McCall 79] e está baseado nos seguintes conceitos:

- *objetivos de qualidade* : são propriedades gerais que o produto deve possuir;
- *fatores de qualidade*: são atributos que determinam a qualidade do produto sob o ponto de vista de seus diferentes usuários;
- *critérios*: são atributos primitivos possíveis de serem avaliados;
- *processos de avaliação*: determinam as métricas a serem usadas de forma a se medir o grau de presença, no produto, de um determinado critério;
- *medidas*: indicam o grau de presença, no produto, de um determinado critério;

- *medidas agregadas*: indicam o grau de presença, no produto, de um determinado fator.

Os objetivos de qualidade são atingidos através dos fatores de qualidade, que podem ser compostos por sub-fatores e são avaliados através de critérios. Os critérios definem atributos de qualidade para os fatores. Medidas são valores resultantes da avaliação de um produto segundo um critério específico.

Objetivos e fatores não são diretamente mensuráveis e só podem ser avaliados através de critérios. Um critério é um atributo primitivo, isto é, um atributo independente de todos os outros atributos. Nenhum critério isolado é uma descrição completa de um determinado fator ou sub-fator. Da mesma forma, nenhum fator define completamente um objetivo. A Figura 2.3 mostra a estrutuda deste modelo.

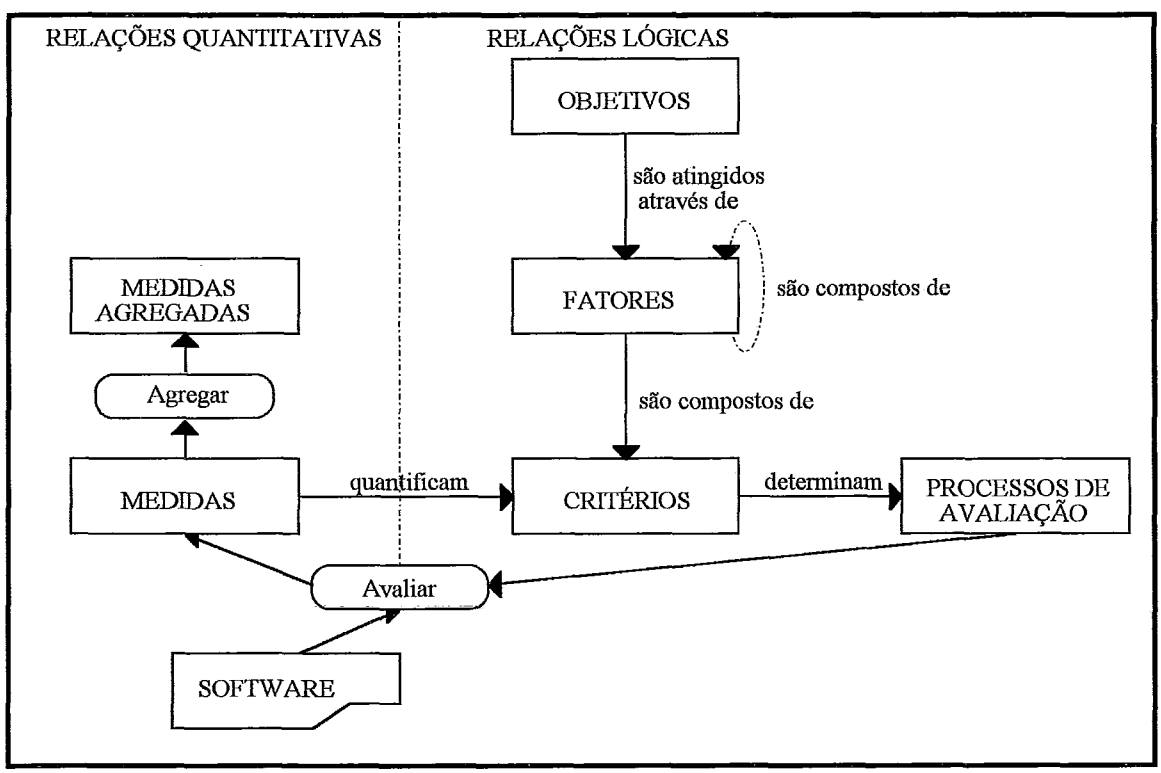

Figura 2.3 - Estrutura do Modelo para Avaliação da Qualidade de Software [Rocha 87]

Inicialmente [Rocha 83], [Clunie 87], o modelo foi utilizado para definir a qualidade e permitir a avaliação de especificações de requisitos. Posteriormente, o trabalho foi estendido considerando as características de especificações de projeto [Passos 92], [Passos 91] e código [Cardoso 87], [Belchior 92a], [Andrade 91] e as particularidades

de diversas áreas de aplicação: software científico [Palermo 89], [Bahia 92], [Commerlatto 94], financeiro [Belchior 92b], educacional [Campos 94a], [Campos 94b].

Esta tese considera os trabalhos anteriores acima referenciados e particulariza o modelo proposto por Rocha para sistemas especialistas, identificando características de qualidade e procedimentos para avaliação de produtos intermediários e do produto final (Capítulo 4).

2.3.3 Métricas de Software

Processos de avaliação ou métricas são utilizados para permitir a quantificação do grau em que os atributos estão presentes em um determinado produto de software de acordo com um modelo de medição especificado.

Na literatura, muito se tem discutido sobre o significado de métricas e medidas. Fenton [Fenton 91] define medição como o processo pelo qual números ou símbolos são atribuídos aos atributos de entidades do mundo real de tal forma que as descreva de acordo com regras claramente definidas. Para que seja possível a medição, portanto, devemos conhecer quais as entidades que estão sendo analisadas e os atributos da entidade que se deseja quantificar.

Entendemos, portanto, métricas como medidas numéricas ou simbólicas que servem para indicar a adequação do software às características de qualidades às quais estão relacionadas [Shepperd 92], [Fenton 91].

Existem várias classificações para métricas na literatura: métricas objetivas e subjetivas [Möller 93], diretas e indiretas [Fenton 91], [Pressman 92] e de produto e processo [Ince 90].

Na classificação de Möller [Möller 93] métricas objetivas são facilmente quantificadas e medidas através de expresões numéricas ou representações gráficas de expressões numéricas que podem ser computadas de documentos de software tais como código fonte, projeto, dados de teste, entre outros. As métricas subjetivas são menos quantificáveis, sendo medidas relativas baseadas em estimativas pessoais ou de grupo e geralmente obtidas por classes de resposta como *excelente*, *regular*, *bom*, *ruim*.

Métricas podem ser medidas diretas ou indiretas. Medidas diretas são aquelas que não dependem da medida de outro atributo, permitindo a quantificação de um fator observado no produto. As medidas indiretas de um atributo envolvem as medidas de um

ou mais atributos relacionados a este [Fenton 91]. Ince [Ince 90] usa uma classificação semelhante a esta denominando de métricas de predição e de resultado e faz uma associação com sua classificação de métricas de produto e processo.

Uma métrica de software é de produto, quando o valor é obtido de algum documento ou código de programa; e métrica de processo, quando o valor descreve o processo de software. Com isso, a métrica de predição é, geralmente, considerada uma métrica de produto que pode ser usada para predizer uma outra métrica, denominada de métrica resultante, que é, normalmente, uma métrica de processo. Utilizar as características de uma especificação do sistema para predizer a quantidade de recursos que serão necessários num projeto de software é um exemplo de métrica de produto (especificação do sistema) sendo utilizada para predizer uma métrica resultante (recursos de um projeto) .

Möller [Möller 93] propõe que para definir e selecionar métricas que são mais aplicáveis para um ambiente de desenvolvimento específico devem ser analisadas suas características organizacionais e técnicas. As características organizacionais referem-se a aplicação de métricas dentro de um ambiente organizacional sendo observados vários aspectos. Um plano de aplicação das métricas deve ser desenvolvido para áreas cujos resultados iniciais tragam o melhor retorno de investimento. Deve ser garantida a aplicação das métricas em todo o ciclo de desenvolvimento do software e as métricas devem ser utilizadas como uma ferramenta de qualidade no gerenciamento do projeto e como uma base para melhorias no processo de desenvolvimento. Além disso, a organização deve incentivar, fiscalizar e dar suporte para o uso de métricas, salientando que as métricas não devem ser uma determinação política, para que não tenha impactos morais negativos, procurando mostrar que as métricas não são para avaliar o desempenho individual dos membros da equipe e sim para melhorar o processo de desenvolvimento.

As características técnicas aplicam-se à própria definição da métrica. Deve ser usado um número limitado de métricas, que sejam fáceis de determinar, cujos dados necessários estejam facilmente disponíveis, e que sejam submetidas a experimentações. Além disso, deve-se ter um compromisso com padronizações da organização para as métricas.

Embora o uso de métricas seja ainda recente, elas podem ser utilizadas para diferentes fins, como por exemplo para estimativa de recursos de projeto, como mecanismo para avaliar o desempenho do projeto e da equipe de desenvolvimento, para avaliação de métodos de desenvolvimento e para ajudar a equipe de desenvolvimento a conseguir boas estimativas da qualidade do seu trabalho.

As principais preocupações da indústria e dos pesquisadores de métricas para software referem-se à falta de validações empíricas de métricas, principalmente métricas para projeto e especificações, à falta de bons experimentos para essas validações e à falta de ferramentas de suporte [Ince 90].

2.4 Gerência da Qualidade de Software

Ao longo do processo de desenvolvimento de software é necessária a existência de uma gerência da qualidade para definir, planejar e controlar as atividades a serem realizadas com o objetivo de garantir a qualidade do produto [Arthur 93]. É, também, função desta gerência, organizar as equipes necessárias para realização dessas atividades e semear a importância da qualidade na organização. Esta última implica em criar uma cultura na organização com relação à qualidade evidenciando seus benefícios [Frewin 86], [Jack 93], [Sefcik 94], e de que o enfoque que se busca é na avaliação do produto e não das pessoas envolvidas em seu desenvolvimento [Möller 93].

A cultura da qualidade implica na atenção à satisfação do usuário, ênfase na melhoria contínua do processo de desenvolvimento, capacitação das equipes e compromisso de todos com o gerenciamento [Arthur 93], [Sanders 94].

É, ainda, responsabilidade da gerência da qualidade informar dos padrões e procedimentos a serem realizados, controlar e fiscalizar a realização desses procedimentos e garantir a qualidade por processos de auditoria e certificação da qualidade alcançada pelo produto a serem realizadas por equipes específicas para esse fim [Rook 86]. Além disso, é necessário rever e avaliar, constantemente, os resultados da política de qualidade para poder melhorar o processo.

Controle da qualidade de software é o conjunto planejado e sistemático de todas as ações necessárias para fornecer uma confiança adequada de que o item ou produto está de acordo com os requisitos técnicos estabelecidos [ANSI/IEEE Std 730 -93]. Gerenciar a qualidade de software implica no planejamento e na realização do controle desta qualidade.

2.4.1 Planejamento do Controle da Qualidade

Quando falamos de qualidade de software, estamos nos referindo a todos os produtos gerados durante o desenvolvimento, sejam especificações ou o produto final.

Dessa forma o controle da qualidade deve ser feito desde o início do desenvolvimento [Frewin 86], [Rocha 87].

O planejamento do controle de qualidade para um projeto implica, primeiramente, na identificação das caracteríticas de qualidade de interesse para o produto a ser desenvolvido, no estabelecimento da importância de cada uma destas características para a satisfação do usuário, e na identificação das relacões e possíveis conflitos entre elas.

Para ser possível realizar avaliações no produto, deve-se, ainda, definir processos de avaliação (ou métricas) para cada característica identificada como relevante para o produto e especificar o grau em que se deseja alcançar a característica.

Com essas definições, são estabelecidos marcos e pontos de controle onde os produtos gerados ao longo do desenvolvimento serão avaliados. É, então, gerado o Plano de Controle da Qualidade que deve conter a descrição de todos os procedimentos a serem adotados, no projeto, para controle da qualidade ao longo do desenvolvimento e avaliação da qualidade do produto final, além de definir a equipe de controle da qualidade. A partir daí, a tarefa de controlar se os requisitos de qualidade estão sendo atingidos é uma atividade presente em todo o ciclo de vida do produto de acordo com o Plano de Controle da Qualidade. A Figura 2.4 mostra um exemplo de roteiro para este plano [ANSI/IEEE Std 730 - 93].

A seguir detalharemos alguns dos elementos que devem ser considerados na elaboração do Plano.

2.4.1.1 Técnicas para Controle da Qualidade de Software

Avaliações para controle da qualidade de software devem estar presentes ao longo de todo o ciclo de vida com o objetivo de assegurar que os requisitos estabelecidos podem ser alcançados, identificar os requisitos que não podem ser alcançados, garantir que o software está sendo representado de acordo com padrões pré-definidos, assegurar que o software é desenvolvido de forma uniforme, descobrir erros para tomar medidas corretivas o mais cedo possível e tornar o projeto mais gerenciável [Pressman 92].

As avaliações incluem processos de *verificação*, para assegurar a correção e consistência no que diz respeito aos produtos de cada fase e às normas fornecidas como entrada para a referida fase; *validação* para avaliar a adequação do produto de software aos seus propósitos, assegurando que o mesmo atenda aos requisitos especificados e *teste* com execução do código para produzir resultados a serem analisados. [Rook 86]

Plano de Controle da Qualidade de Software

1. Proposta
2. Referência de Documentos
3. Gerência
   - 3.1 Organização
   - 3.2 Tarefas
   - 3.3 Responsabilidades
4. Documentação
   - 4.1 Proposta
   - 4.2 Requisitos Mínimos de Documentação
     - 4.2.1 Especificação de Requisitos de Software
     - 4.2.2 Descrição do Projeto de Software
     - 4.2.3 Plano de Verificação e Validação de Software
     - 4.2.4 Relatório de Verificação e Validação de Software
     - 4.2.5 Documentação do Usuário
     - 4.2.6 Plano de Gerência de Configuração de Software
   - 4.3 Outras Documentações
5. Padrões, Práticas, Convenções e Métricas
   - 5.1 Proposta
   - 5.2 Conteúdo
6. Revisões e Auditorias
   - 6.1 Proposta
   - 6.2 Requisitos Mínimos
     - 6.2.1 Revisão de Projeto Preliminar
     - 6.2.2 Revisão de Projeto Crítico
     - 6.2.3 Revisão do Plano de Verificação e Validação
     - 6.2.4 Auditoria Funcional
     - 6.2.5 Auditoria Física
     - 6.2.7 Auditoria no Processo
     - 6.2.8 Revisões Administrativas
     - 6.2.9 Revisão do Plano de Gerência de Configuração de Software
   - 6.3 Outras Revisões e Auditorias
7. Teste
8. Relatório de Problemas e Ações Corretivas
9. Ferramentas, Técnicas e Metodologias
10. Controle de Código
11. Controle de Comunicação
12. Controle de Fornecedores
13. Registros, Manutenção e Retenção
14. Treinamento
15. Gerência de Riscos

Figura 2.4 - Exemplo de Roteiro do Plano de Qualidade [ANSI/ IEEE Std 730 - 93]

As avaliações devem ser realizadas utilizando-se técnicas para controle da qualidade. O uso dessas técnicas deve ser adequadamente planejado e controlado para que se atinja os objetivos propostos para a avaliação. Essas técnicas apresentam diversos graus de formalidade e são adequadas a diferentes tipos de projeto [Boegh 93]. Na literatura encontramos muitas dessas técnicas: modelos de confiabilidade [Musa 87], o Cleanroom [Selby 87], prova formal [Dyer 87], *walkthrough* [Freedman 84], [Yourdon 89], inspeção [Fagan 76], inspeção por fases [Knigh 93] e testes de software [Pressman 92], [Coward 88]. A seguir abordaremos, em mais detalhes, algumas destas técnicas.

2.4.1.1.1 Walkthrough

A técnica de *walkthrough* envolve atividades de planejamento, preparação e uma reunião da qual participam três a cinco pessoas, devendo ter uma duração em torno de duas horas. O planejamento envolve, assim que a equipe de desenvolvimento der por concluído um produto que deverá ser avaliado, o agendamento da reunião e a definição dos participantes. É de responsabilidade do gerente da qualidade, que realiza esta atividade. O material a ser avaliado é distribuído para os avaliadores se prepararem para a reunião, lendo o material e tomando nota dos problemas identificados.

Participam da reunião um moderador, representantes da equipe de desenvolvimento e dos usuários, e avaliadores externos ao projeto. Um representante dos desenvolvedores assume o papel de narrador e outro o de secretário.

Durante a reunião, é feita uma narrativa do documento dando-se espaço para que cada participante faça suas observações, que são discutidas pelo grupo. Ao se identificar um problema sobre o qual existe consenso, o secretário toma nota do mesmo para posterior correção.

No final da reunião, os participantes decidem-se pela aceitação do produto sem alterações, aceitação com as alterações determinadas ou pela rejeição devido à presença de problemas graves. Neste caso, uma vez corrigidos os problemas, deve-se submeter o produto a outro *walkthrough*. Tem-se como produto da reunião, um laudo com a lista de todos os problemas detectados e informações consideradas relevantes (identificação do produto, data da reunião, nome dos participantes, etc).

2.4.1.1.2 Inspeção

Nos últimos anos algumas organizações tem substituído suas técnicas de avaliação pela técnica de inspeção proposta por Fagan em 1976, por a considerarem superior às outras [Ackerman 89]. A técnica de inspeção é semelhante ao *walkthrough* embora seja mais formal por ser baseada em critérios, previamente definidos, para avaliação das características de qualidade.

A realização de uma inspeção implica, além das etapas previstas na técnica de *walkthrough* (planejamento, preparação e reunião), uma etapa de apresentação. Nesta etapa os desenvolvedores fazem um tutorial, de aproximadamente trinta minutos, sobre o material a ser inspecionado destacando o que deve ser analisado, facilitando, com isso, a preparação individual dos inspetores.

Na preparação individual, os inspetores devem avaliar o material de acordo com a lista de critérios previamente estabelecidos.

Na reunião, o narrador deve procurar ser objetivo, sintetizando cada seção do material. O moderador deve conduzir a reunião procurando manter a objetividade e o cronograma. Deve, ainda, garantir que sejam considerados todos os critérios estabelecidos. No final da reunião, a lista de critérios deve ter sido completamente analisada e com respostas consensuais a todas as questões. Deve-se decidir, assim como na técnica de *walkthrough*, pela aceitação ou não do produto, definindo se deve ou não haver uma re-inspeção. Os participantes da reunião são os mesmos previstos na técnica de walkthrough.

Experiências comprovam que o uso de inspeções contribui significativamente para a melhoria da qualidade do produto bem como para a redução do custo relacionados a testes, embora não elimine a necessidade de realizá-los [Ackerman 89].

2.4.1.1.3 Inspeção por fases

Inspeção por fases [Knight 93] é uma variação da técnica de inspeção. É uma técnica disciplinada e rigorosa onde podem ser avaliados todos os produtos gerados ao longo do desenvolvimento. A avaliação do produto é feita através de uma série de avaliações parciais (*fases*). Cada avaliação parcial tem por objetivo verificar se o produto possui uma ou mais propriedades. As propriedades examinadas são organizadas na inspeção por fases, de forma que cada avaliação parcial pode assumir a existência das propriedades verificadas ou validadas nas avaliações parciais anteriores. A técnica de inspeção por

fases serve, portanto, não só para a detecção de erros como, também, para a garantia da qualidade de acordo com os requisitos de qualidade identificados no início do projeto.

Numa inspeção por fases, as avaliações parciais podem ser realizadas com *inspetores individuais* ou com *múltiplos inspetores*. Uma avaliação parcial, com inspetores individuais, é um processo rigoroso e controlado por uma lista de questões a serem respondidas pelo avaliador. Estas questões são métricas e/ou indicadores do grau em que foram atingidos os atributos de qualidade considerados na avaliação e relevantes naquele momento. Uma avaliação parcial, com múltiplos inspetores, é realizada para analisar questões subjetivas e onde se busca, através de uma reunião, obter o consenso dos avaliadores. O processo de realização de uma avaliação com múltiplos inspetores consiste de duas etapas: inicialmente uma *avaliação individual* onde cada inspetor analisa o material preparando-se para a segunda etapa que é a *avaliação em grupo*. O procedimento para realização de reuniões de inspeção é o tradicional.

2.4.1.1.4 Testes

Os testes representam uma atividade crítica para garantia da qualidade, pois significam de certa forma, uma avaliação "visível" da qualidade do produto que está sendo construído. A avaliação, ao contrário das técnicas anteriores, é feita através da submissão do programa à execução.

De fato, os testes representam uma etapa do processo de desenvolvimento onde tem-se um objetivo destrutivo. Segundo Myers[1] [in Pressman 92] um bom caso de teste é aquele que tem uma alta probabilidade de identificar erros e um teste bem sucedido é aquele que evidencia erros ainda não identificados.

Testes podem ser divididos em duas categorias: funcional e não-funcional. Testes funcionais são empregados quando se deseja testar um novo programa, ou um programa que foi modificado, para verificar se produz o resultado correto. Refere-se, portanto, à implementação da função solicitada pelo usuário. No entanto, mesmo que a implementação das funções solicitadas pelo usuário produza resultados corretos não significa necessariamente que todos os requisitos do software estão satisfeitos. Os testes não-funcionais referem-se à avaliação de que o software atende a obrigações legais, tem um desempenho dentro do que foi especificado, está escrito de acordo com normas e padrões estabelecidos e outros requisitos solicitados pelo usuário. [Coward 88]

[1] G. Myers; "The Art of Software Tests"; Wiley; 1979

A atividade de testes, para ser produtiva, deve ser planejada antecipadamente e conduzida de forma sistemática realizando-se uma avaliação progressiva do produto. Inicialmente, deve ser realizado o teste de cada módulo individualmente para garantir que estes funcionam apropriadamente como uma unidade do programa (*testes de unidade*). A seguir, faz-se o *teste de integração* das unidades verificando-se a composição dos módulos para formação do programa como um todo. Depois do software estar integrado, *testes de validação* são realizados para garantir que o sistema construído alcançou os requisitos estabelecidos. Finalmente, é realizado o *teste do sistema* no qual o sistema é avaliado dentro do seu contexto, combinado com outros elementos (como hardware, banco de dados, pessoas, etc), verificando, assim, que todos os elementos se adequam apropriadamente e que todos os requisitos do sistema foram alcançados. [Pressman 92]

2.4.1.1.5 Prova Formal

A verificação formal consiste em tratar o programa como um objeto matemático que possa ser avaliado em conjunto com suas especificações [Hoare 78]. Segundo Pressman [Pressman 92] se a especificação de requisitos e a linguagem de programação puderem ser repreentadas de uma maneira (tanto sintática quanto semântica) rigorosa, pode ser possível aplicar provas matemáticas de correção para demonstrar que o programa equivale a sua especificação.

Tentativas de provar a correção de programas não são novas [Dijkistra 76], [Linger 79]. Um programa é visto como uma sequência de instruções que implementa alguma função especificada. Em vários pontos desta sequência pode ser possível determinar, a partir da especificação, o valor correto das variáveis do programa, o status apropriado do controle de informação e outras relações internas. Portanto, para as instruções selecionadas no programa s_1, s_2, ..., s_n pode ser possível avaliar se as condições específicas c_1, c_2, ..., c_n são verdadeiras. Para provar a correção de programas as instruções entre as instruções selecionadas s_i e s_{i+1} devem garantir causar que a condição especificada c_i seja transformada em c_{i+1}. Progressivamente pode ser garantido que a aplicação do programa para uma entrada irá produzir a condição de saída especificada. [Pressman 92]

Segundo Pressman [Pressman 92], alguns pesquisadores de prova formal acreditam que a abordagem descrita acima, ou outras variações desta, devem ser aplicadas pelos desenvolvedores para projeto e implementação de módulos de programas. No entanto,

outros sentem que a única esperança para a utilização da prova formal está no desenvolvimento de ferramentas automatizadas.

2.4.1.2 Controle Estatístico da Qualidade de Software

O controle estatístico da qualidade reflete um tendência no mercado se tornar mais quantitativo sobre qualidade. Pressman [Pressman 92] propõe que para realizar o controle estatístico da qualidade, a informação sobre os defeitos do software devem ser coletadas e classificadas e, são atribuídas a um conjunto de causas como: especificação incorreta ou incompleta, entendimento errado na comunicação com o usuário, não correspondência com especificações, testes errados ou incompletos, módulo de interface inconsistente, entre outros. A partir daí deve ser determinado o número total de defeitos para cada causa relacionada, e deste total, o número de defeitos considerados sérios, moderados ou triviais, construindo uma tabela com estes dados.

Segundo Pressman, os desenvolvedores de software podem calcular um índice de defeito (ID) para cada fase principal do processo de desenvolvimento. Assim, depois da análise, projeto, codificação, teste e manutenção deve ser determinando o número total de defeitos (Di), o número de defeitos sérios (Si), o número de defeitos moderados (Mi), o numéro de defeitos triviais (Ti), o tamanho do produto (TM em linhas de código, instruções de projeto, páginas de documentação) e a determinação de um peso para os defeitos sérios, moderados e triviais (w_j, j = 1 até 3). É então calculado o indíce para cada fase (IFi = w_1 (Si/Di) + w_2 (Mi/Di) + w_3 (Ti/Di)). O índice de defeito (ID) é calculado pelo efeito acumulativo de cada IFi, da seguinte forma: ID = $\Sigma(i \times IFi) / TM$.

O índice de defeito é usado em conjunto com as informações coletadas na tabela para determinar uma indicação de melhorias na qualidade do software.

2.5 Conclusão

Melhorar a qualidade de software é uma meta fundamental no mercado de software atual. Para que isso seja possível, é importante a gerência da qualidade, o uso de técnicas adequadas para especificar e avaliar tanto a qualidade do produto de software quanto a qualidade de seu processo de desenvolvimento.

Neste capítulo discutimos a questão da qualidade de software considerando o atual estado da arte. No próximo capítulo iremos abordar a questão da qualidade de sistemas especialistas, mais especificamente o estado da arte no que se refere à validação e verificação destes sistemas.

Capítulo 3
Sistemas Especialistas: Conceituação, Verificação e Validação

3.1 Introdução

Qualidade é um importante fator para o sucesso da tomada de decisão realizada por um sistema e, consequentemente, para o seu uso. Um sistema especialista que não seja verificado e validado apropriadamente pode tomar decisões não adequadas. Diante dessa preocupação muito se tem discutido sobre avaliação da qualidade de sistemas especialistas.

Neste capítulo, vamos descrever conceitos, métodos, técnicas e ferramentas apresentados na literatura sobre verificação e validação de sistemas especialistas.

3.2 Sistemas Especialistas: Conceituação

Sistemas especialistas representam uma conquista da área de Inteligência Artifical, na busca de programas de computador que solucionassem problemas de uma forma que seria considerada inteligente se realizada por especialistas humanos. Os programas de Inteligência Artificial exibem um comportamento inteligente através da aplicação de heurísticas e não do processo algoritmico dos programas convencionais. Para tornar um programa inteligente, ele deve ser provido de alta qualidade e de um conhecimento específico sobre uma área do problema [Waterman 86].

Segundo Waterman o conhecimento necessário para o sistema é organizado separando o conhecimento sobre o domínio do problema e o conhecimento sobre a solução do problema. *Sistemas baseados em conhecimento* organizam o conhecimento dessa forma. Portanto, os sistemas especialistas, segundo Waterman, são sistemas baseados em conhecimento que aplicam um conhecimento especializado de um domínio de aplicação (Figura 3.1).

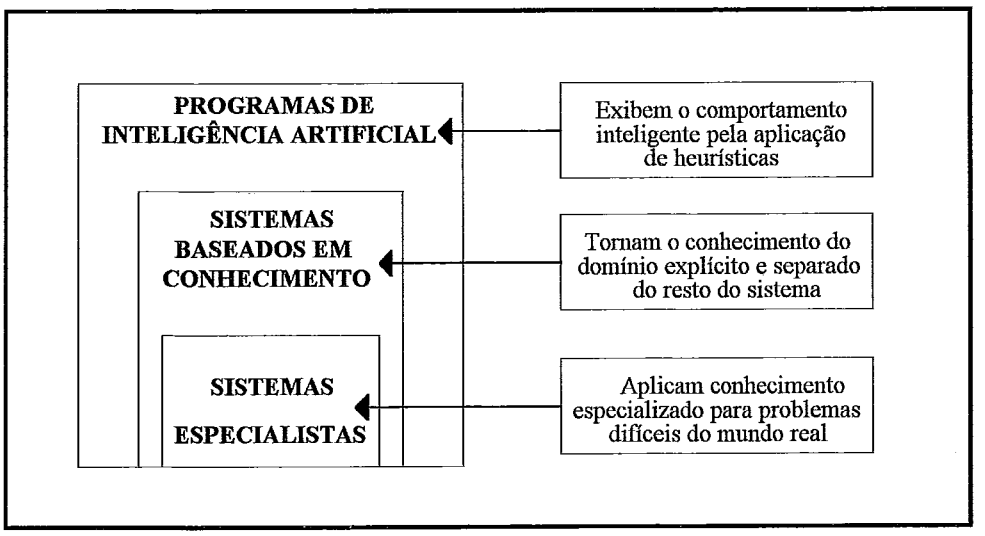

Figura 3.1 - Classificação de sistemas de IA [Waterman 86]

Meseguer [Meseguer 93] define sistemas especialistas como programas responsáveis pela solução de problemas de domínios especializados nos quais a solução efetiva do problema normalmente requer um especialista humano, considerando, portanto, um outro fator para sistemas especialistas: o especialista.

O conhecimento do domínio do problema armazenado é chamado base de conhecimento e consiste de uma coleção de itens de conhecimento. Cada item de conhecimento representa uma unidade de conhecimento do domínio de aplicação. As relações entre estes itens de conhecimento permitem deduções que são as soluções para os problemas do domínio da aplicação. A implementação da solução do problema é denominado mecanismo de inferência [Morell 88].

A máquina de inferência é a chave para a operação de um sistema especialista [Geissman 88]. É ela quem interpreta a base de conhecimento para solucionar os problemas do domínio da aplicação. A forma como é estruturada depende da natureza do domínio do problema e da maneira pela qual o conhecimento é representado e organizado no sistema especialista. Linguagens mais poderosas para construção de sistemas especialistas já oferecem a máquina de inferência embutida; outras, requerem que ela seja implementada [Waterman 86].

A função de qualquer esquema de representação do conhecimento é capturar as características essenciais de um domínio do problema e produzir informações acessíveis aos procedimentos de solução do problema [Luger 89]. Nesse sentido são utilizados dois paradigmas de Inteligência Artificial: o simbolista que considera a inteligência apenas no comportamento do indivíduo e considera não haver inteligência relacionada a máquina,

e, o conexionista que determina uma estrutura de rede de elementos computacionais, as redes neurais, sobre as quais está a inteligência.

O desenvolvimento de sistemas especialistas normalmente utiliza o paradigma simbolista. Entretanto, existem sistemas especialistas híbridos que representam o conhecimento através da utilização dos dois paradogmas. A representação do conhecimento pode ser caracterizada como sendo orientada a fatos ou a outros conhecimentos que possam ser usados no raciocínio e implementados por uma linguagem de programação [Duarte 91]. Os fatos são verdades relevantes a algum domínio do problema e são o que se quer representar; e a representação dos fatos é um formalismo expresso em estruturas de conhecimento que podemos manipular.

As estruturas de representação do conhecimento podem ser classificadas em quatro categorias [Luger 89]: *lógica*, que usa expressões da lógica formal para representar o conhecimento e aplica regras de inferência e procedimentos de prova formal a instâncias do problema, como o cálculo de predicados de primeira ordem; *procedural*, que para representar o conhecimento utiliza um conjunto de instruções de solução do problema, como por exemplo as regras de produção; *redes* que representa o conhecimento como um grafo no qual os nós representam objetos ou conceitos do domínio do problema e os arcos as relações entre eles, como por exemplo as redes semânticas, e; *estruturada*, que estende o esquema de redes permitindo que cada nó seja uma estrutura de dados complexa como por exemplo *frames*, e objetos.

Uma das formas mais antigas de representar o conhecimento é a lógica de primeira ordem ou cálculo de predicados [Luger 89], que provê uma sintaxe bem definida e uma semântica clara. Na lógica de primeira ordem uma sentença em linguagem natural é traduzida em estruturas simbólicas compreendidas de *conectivos lógicos* (negação ($\sim$), conjunção (&, $\wedge$), disjunção ($\vee$), implicação ($\rightarrow$) e equivalência ($\leftrightarrow$)); *quantificadores* (universal ($\forall$) e existencial ($\exists$)); *constantes* (letras minúsculas do início do alfabeto); *variáveis* (letras minúsculas do final do alfabeto); *funções* ($f(t_1, t_2, ..., t_n)$, $n \geq 0$ e t_i são constantes, variáveis ou funções definidos para algum domínio) e *predicados* ($P(t_1, t_2, ..., t_n)$, $n \geq 0$ e t_i são constantes, variáveis ou funções definidos para algum domínio) [Patterson 90].

As regras de produção são o tipo mais popular de técnica de representação do conhecimento [Waterman 86], sendo constituídas de duas partes: a primeira parte (lado esquerdo da regra) é uma declaração com prefixo se (*if*) sendo chamada de antecedente e que expressa uma situação, premissa ou condição, e a segunda parte (lado direito da regra) tem prefixo então (*then*) sendo o consequente e expressa uma ação em particular, conclusão, ou resposta aplicada quando a situação ou premissa é verdadeira. As regras

de produção são fáceis de serem criadas e entendidas por serem semelhantes a forma como é armazenado o conhecimento humano, mas apresenta como desvantagem a necessidade de pesquisa sequencial na base de conhecimento, comprometendo o sistema em relação ao tempo de execução de uma pesquisa quando há um grande número de regras [Luger 89], [Patterson 90].

As redes semânticas foram originalmente desenvolvidas para uso como modelo psicológico da mente humana mas seu método de representação foi estendido para Inteligência Artificial [Waterman 86]. Nesta representação o conhecimento é organizado em grupos semânticos coerentes [Patterson 90]. Uma rede semântica consiste de *nós*, que representam objetos, conceitos ou eventos, e são conectados ligações chamadas *arcos* que descreve as relações entre os nós. Os arcos podem ser definidos de diferentes formas dependendo do tipo de conhecimento sendo representado. Esta representação é adequada para representar hierarquia de taxonomias, na qual os objetos ou conceitos são representados através da conexão de herança entre os nós que são os objetos de repesentação. Os arcos mais comuns para representar hierarquias são o *é_um* e *é_parte_de* [Waterman 86], [Duarte 91].

Frames é uma estrutura de dados para representação do conhecimento existente de objetos, eventos e situações estereotipadas [Luger 89]. Um *frame* é organizado como uma rede semântica [Waterman 86], pois pode ser considerada como uma rede de nós e relações organizadas em hierarquia, na qual os nós superiores representam conceitos gerais e os inferiores instâncias específicas desses conceitos. O conceito para cada nó é definido por uma coleção de atributos e valores desses atributos. Esses atributos são chamados de *slots* e podem conter informações sobre valores padrões, condições de preenchimento, ponteiros para outros *frames* relacionados e procedimentos que são ativados quando necessário por diferentes propósitos [Patterson 90].

A abordagem de agrupar conhecimento e procedimentos relacionados em uma única unidade levou a definição da representação de objetos, que, apresenta, portanto, alguma similaridade com *frames*. Na representação orientada a objetos a relação entre dados e procedimentos é o inverso do tradicional, isto é, os dados são objetos principais, e os procedimentos (chamados de métodos) são incorporados ao objeto que não pode ser acessado por procedimentos externos. Através desta estrutura, chamada de encapsulação de dados, objetos são associados com seus próprios procedimentos, sendo responsáveis por suas próprias ações. O objeto possui um nome, uma caracterização de classe, alguns atributos e um conjunto de operações, sendo qua a interação com outros objeto se faz somente através da troca de mensagens. A utilização de objetos tem sido considerada muito útil para representar situações do mundo real na qual os objetos reais podem ser

modelados numa correspondência direta com classes e objetos de programas. [Patterson 90]

Representações híbridas, utilizando duas dessas representações descritas, são as mais frequentemente utilizadas em sistemas de maior porte [Garcia 91], principalmente incorporando regras com *frames* ou objetos.

Embora o paradigma conexionista não esteja no mesmo nível de desenvolvimento do simbolista, tem sido realizada muitas pesquisas por representar uma perspectiva de oferecer certas características interessantes para a representação do conhecimento como o aprendizado automático e processamento paralelo. As redes neurais são redes grandes com elementos ou nós simples que processam a informação dinamicamente em resposta as entradas externas. Os nós são modelos simplificados de neurônios e o conhecimento está distribuído ao longo da rede na forma de conexões entre os nós e ligações com pesos que formam as entradas dos nós. As ligações com pesos servem para inibir ou exercitar os valores estimulados na entrada que serão adicionados ao conjunto de nós. Os modelos de redes neurais apresentados na litertura são caracterizados por três elementos básicos: um conjunto de unidades de processamento, uma topologia de interligação (na qual incluem a inteligência) e um esquema de aprendizado [Carvalho 89], [Luger 89], [Patterson 90], [Duarte 91].

3.3 Características dos Sistemas Especialistas

Outra forma de caracterizar sistemas especialistas é comparando-os com outros sistemas. Sistemas especialistas são substancialmente diferentes de outros sistemas.

Sistemas especialistas diferem dos sistemas convencionais, por exemplo, pelo seu caráter não procedural, pela existência de uma base de conhecimento e pela sua forma recursiva de desenvolvimento em oposto à linha tradicional de desenvolvimento de sistemas convencionais [Naser 88]. Segundo Waterman [Waterman 86], a diferença básica entre estes sistemas é que os sistemas especialistas manipulam conhecimento enquanto que os sistemas convencionais manipulam dados. Meseguer [Meseguer 93] refere-se às diferenças entre sistemas convencionais e sistemas especialistas de acordo com quatro pontos de vista: o tipo de problema considerado, os métodos de solução utilizados, as linguagens e estilos de programação empregadas e o uso efetivo de ambos os sistemas.

Segundo este autor a principal diferença entre sistemas especialistas e sistemas convencionais refere-se ao tipo de problema considerado para cada um deles. Os

problemas tratados pelos sistemas convencionais são bem definidos, permitem a clara decomposição em subpartes e trabalham com entradas bem definidas e completas, produzindo saídas precisas. No entanto, os sistemas especialistas solucionam problemas resolvidos por especialistas humanos, sendo mal estruturados e tem que lidar com informações incompletas, incertas ou até mesmo inconsistentes. Além do mais, um problema pode ter múltiplas soluções igualmente aceitáveis.

Estas diferenças são refletidas no grau de precisão da especificação do sistema. Nos sistemas convencionais há uma completa tradução dos requisitos funcionais em suas especificações, o que não é possível no caso dos sistemas especialistas. Na prática, verifica-se que nos sistemas especialistas existem requisitos do usuário que não podem ser expressos em termos formais numa linguagem de especificação. Embora uma completa tradução dos requisitos do usuário em especificações não seja possível, uma tradução parcial pode ser conseguida. Isto sugere dividir os requisitos do usuário em duas classes: requisitos totalmente formalizáveis e requisitos parcialmente formalizáveis [Meseguer 93], [Hoppe 93]. Um requisito é totalmente formalizável se pode ser completamente traduzido em especificações (como a correção da base de conhecimento). Da mesma forma, um requisito é parcialmente formalizável se ele não pode ser completamente traduzido em especificações, como, por exemplo, no caso de sistemas médicos o requisito de gerar diagnósticos aceitáveis para todos os casos típicos. Não existe uma boa definição de qual é um diagnóstico aceitável para um caso, sendo, algumas vezes, um problema de preferência pessoal. Assim, a falta de uma definição precisa para este conceito implica que este requisito não pode ser completamente traduzido em especificações. Os requisitos parcialmente formalizáveis são, por natureza, mais difíceis de serem validados.

Outra grande diferença entre os sistemas especialistas e sistemas convencionais é a abordagem de solução dos problemas. Os sistemas convencionais são solucionados por algoritmos que podem ser provados estarem corretos independentes de qualquer implementação. Obviamente isto não garante a validade da implementação já que erros podem ser introduzidos quando o algoritmo é codificado. No caso de sistemas especialistas, os problemas são solucionados por associações heurísticas que compõem modelos que simulam o comportamento de especialistas humanos (baseado em experiências pessoais) para um problema do domínio. Dessa forma, uma validação efetiva é adiada até a fase de implementação.

A solução gerada está ligada diretamente com a programação. Os programas dos sistemas convencionais são desenvolvidos usando linguagens procedurais enquanto que os sistemas especialistas usam linguagens declarativas que são interpretadas por

máquinas de inferência. Os estilos de programação nos sistemas convencionais e nos sistemas especialistas são totalmente diferentes: os programas dos sistemas convencionais lidam com variáveis, condições, *loops* e procedimentos; ao contrário dos sistemas especialistas que lidam com valores verdade, dependências de regras e associações heurísticas.

Ao analisar as diferenças no uso efetivo de sistemas especialistas e convencionais, Messeguer considera dois aspectos: o nível organizacional, que analisa o uso dos softwares na organização e o nível do usuário, que analisa como o usuário final lida com o software. Do ponto de vista do nível da organização, a tecnologia de sistemas convencionais é totalmente consolidada e comercialmente aceitável. Suas aplicações são, frequentemente, o único caminho possível para processar informações. As mudanças causadas na organização por estas aplicações são aceitas como consequência natural do uso do software. Os sistemas especialistas, ao contrário, não são geralmente embutidos no fluxo de informação das organizações, sendo programas isolados para apoio à decisão, avisos de alerta ou treinamento de novatos. O impacto de suas aplicações nas organizações ainda é um campo a ser pesquisado. Essa diferença tem muita relação com o nível de maturidade e o grau de confiança nos dois tipos de sistema. Do ponto de vista do usuário, existem duas diferenças. A primeira refere-se ao fato de que enquanto as aplicações de sistemas convencionais são, frequentemente, a única forma de processar informações, as aplicações de sistemas especialistas são, geralmente, opcionais e o usuário tem a decisão de usar ou não a solução do sistema especialista. Isso requer adquirir a confiança do usuário para com o sistema especialista, o que normalmente exige que o sistema explique suas deduções e recomendações de forma que o usuário entenda (este fator foi considerado muito importante em pesquisas realizadas no projeto MYCYN). Outra diferença corresponde às responsabilidades legais no uso do software. No caso dos sistemas convencionais o usuário não tem qualquer responsabilidade por falhas de desempenho da execução do software, pois as soluções obedecem a protocolos bem definidos para processamento de informações. Já no caso de sistemas especialistas (por exemplo, sistemas médicos como o MYCIN) o usuário (no caso do MYCIN, médicos) tem toda a responsabilidade sobre suas decisões, incluindo erros induzidos por recomendações erradas do sistema especialista, o que justifica ainda mais a necessidade do sistema especialista ter que conquistar a confiança do usuário.

Existem ainda aspectos técnicos, ambientais e de projeto que distinguem completamente os sistemas especialistas de outros sistemas [O'Leary 87].

Os aspectos técnicos incluem a característica dos sistemas especialistas de processar informações simbólicas e não apenas numéricas e com isso permitir solucionar problemas

não-numéricos que normalmente são menos precisos do que os problemas numéricos; exigir um esquema de representação do conhecimento especial para torná-lo acessível (não necessário nos outros sistemas por não possuírem representação do conhecimento); e serem desenvolvidos com linguagens de Inteligência Artificial que processam informações simbólicas ou utilizando ferramentas de sistemas especialistas (*shells*) que já oferecem uma máquina de inferência e facilitam a entrada do conhecimento.

Dois aspectos ambientais distinguem claramente os sistemas especialistas de outros sistemas. O primeiro, refere-se à característica dos sistemas especialistas influenciarem diretamente na decisão ou mesmo no fato deste tomar decisões. O segundo aspecto, refere-se ao fato de que a especialidade a ser modelada pelo sistema especialista, geralmente, está em pequena quantidade, é um recurso caro ou não disponível em uma localização específica.

A metodologia de projeto também distingue os sistemas especialistas de outros sistemas pois, eles são, normalmente, desenvolvidos pela filosofia *middle-out* em vez da abordagem tradicional de projeto *top-down* ou *bottom-up*. Além disso, os sistemas especialistas evoluem sempre, o processo de tomada de decisão é gradualmente entendido e modelado.

3.4 Sistemas Especialistas: Verificação e Validação

Os termos verificação, validação e avaliação são constantemente confundidos na literatura. Balci e Sargent[1] [in O'Leary 90], em uma revisão da literatura publicada sobre validação, constataram que não existe uma definição padrão para os termos e que mais de dezesseis diferentes termos são utilizados de formas diferentes. Hoppe [Hoppe 93] propõe uma terminologia para os termos a partir da análise de diferentes definições propostas na literatura. Segundo este autor quase todos os autores tem desenvolvidos suas próprias terminologias.

Usaremos estes conceitos no sentido mais tradicional, conforme proposto por Boehm [Boehm 80] e também aceita como terminologia para o caso de sistemas especialistas segundo a análise de Hoppe, em que verificação é a avaliação de que se está construindo correto o produto, enquanto que validação é a avaliação de que se está construindo o produto correto. Ou seja, verificação consiste em garantir que o sistema implementa corretamente sua especificação [O'Keefe 87]; e validação consiste em

[1] O.Balci e R.G.Sargent, "A Bibliography of the Credibility and Validation of Simulation and Mathematical Models", Simuletter, vol. 15, No. 3, 1984

comprovar se o sistema desempenha com um nível aceitável de exatidão [O'Keefe 87], com respeito à tarefa que se propõe [Mengshoel 93] de acordo com as necessidades e requisitos do usuário (totais ou parciais) [Meseguer 93].

Morell [Morell 88] define verificação como o processo de demonstração de que o software possui características especificadas na sua documentação (especificação), enquanto que validação é a demonstração de que o software possui características desejadas pelo usuário final. Com isso, afirma que sem especificação não é possível realizar verificação, mas validação pode ser feita.

3.4.1 Problemas para Verificação e Validação de Sistemas Especialistas

Um obstáculo para a total aceitação de sistemas especialistas é a falta de uma metodologia para verificação e validação que é dificultada pela falta de documentação estável e métodos não muito adequados para a avaliação dos resultados de testes [Green 87], [Naser 88]. Se faz necessário, portanto, abordagens metodológicas e displinadas, além de ferramentas para suporte a estas abordagens [Bench-Capon 93].

Em 1987, Green [Green 87] constatava a existência de um ciclo vicioso dentro das organizações que impedia o desenvolvimento de métodos para avaliação da qualidade de sistemas especialistas: a avaliação de sistemas especialistas não era feita porque ninguém solicitava, ninguém solicitava porque ninguém sabia o quê e nem como avaliar e ninguém sabia avaliar porque ninguém nunca realizou avaliações. Green sugeria que para quebrar este ciclo e conseguir progresso em direção a uma efetiva metodologia de avaliação é necessário realizar experimentações. O estado da arte atual mostra que muito se tem experimentado, na tentativa de alcançar essa metodologia [O'Leary 87], [Schultz 88], [Mengshoel 93], [Meseguer 92]. Com o sucesso de sistemas especialistas, as empresas sentem maior necessidade de processos de avaliação.

Realmente, o estado da arte mostra que ainda não existe uma metodologia universalmente aceita para verificação e validação de sistemas especialistas devido a vários problemas específicos.

Um dos principais problemas é que os requisitos de sistemas especialistas são muito difíceis de serem determinados [Geissman 88], [Meseguer 93], além de serem imprecisos ou de serem alterados rapidamente [Green 87]. Com os sistemas especialistas sendo construídos por refinamentos e interação com especialistas, os requisitos são modificados e, às vezes, cada vez menos registrados.

As técnicas convencionais de acompanhamento da execução através do exame do código para verificar se o sistema implementado satisfaz os requisitos é muito difícil de ser realizado [Green 87], [Geissman 88], [O'Leary 87]. A sequência de execução de um sistema especialista a partir do exame da base de conhecimento pode ser extremamente difícil de se determinar, mesmo com as técnicas tradicionais de prova baseada em comparações de pré-condições/pós-condições [Schultz 88]. A capacidade do sistema especialista produzir suas saídas é uma propriedade da interação entre a base de conhecimento e a máquina de inferência, só podendo ser analisada quando o sistema é executado.

Para analisar os requisitos do sistema no projeto e código é necessário identificar unidades de funções em vários níveis. Um projeto modular, *top-down* e decomposto hierarquicamente pode não ser possível em algumas arquiteturas de sistemas especialistas. Utilizando-se regras é difícil ter uma estrutura modular, embora regras diferentes lidam com casos diferentes. Com *frames* a situação pode ser mais parecida com a estrutura procedural. A orientação a objetos oferece um caminho para alcançar alguma modularidade, mas não existe nenhum padrão definido para projeto. [Schultz 88]

Outro problema é que os sistemas especialistas (especialmente aqueles que trabalham com incerteza ou dados incompletos) podem ter muitos estados possíveis, o que torna a aplicação de testes exaustivos impraticável [Geissman 88].

Além de todas essas dificuldades, não existe métodos para avaliação de resultados de teste de sistemas especialistas aceitáveis e confiáveis. A abordagem de ter especialistas no domínio da aplicação tem alguns inconvenientes. Especialistas podem não estar disponíveis ou podem não ser independentes quando for necessária uma avaliação independente [Green 87].

3.4.2 Características de Qualidade para Sistemas Especialistas

Vários autores identificam características de qualidade a serem avaliadas em sistemas especialistas. Conrath [Conrath 91] utilizando o conceito de avaliação baseado na definição de qualidade da ISO, determinou um conjunto de características de qualidade segundo dois aspectos: técnicos e sociais.

Os aspectos técnicos de um sistema compreendem a avaliação do sistema como um produto tecnológico [Guida 94] no que se refere à tarefa que é desempenhada pelo sistema e à tecnologia através da qual o sistema é implementado [Conrath 91]. Quanto à

tarefa são consideradas as características da tarefa ter sido escolhida para aplicação de sistemas especialistas (dificuldade, escassez e utilidade da tarefa), a adequação da tarefa para sistemas especialistas (confiança, realismo e confiança) e a re-elaboração do trabalho (no que diz respeito à inovação e simplificação do trabalho com sistemas especialistas). Quanto à tecnologia são considerados os critérios de entrada (interação e codificação), processamento (como velocidade, inferência e explanação), saída (no que diz respeito à precisão, especificidade e apresentação) e a interface (amigável, fácil de alterar e compatível com outros sistemas).

Os aspectos sociais referem-se às pessoas envolvidas com o sistema e ao contexto da organização. Em relação às pessoas, Conrath considera os fatores referentes ao impacto do sistema especialista para seus usuários (como o estímulo e a redução no trabalho) e os fatores referentes às pessoas durante o processo de desenvolvimento do sistema (como entusiasmo, qualificação do pessoal e ajuda participativa dos especialistas). Quanto à organização analisa as características organizacionais (como burocracia, treinamento e atualização do aprendizado) e os benefícios econômicos que podem provir dos sistemas especialistas (lucros e aumento de competitividade no mercado).

Guida [Guida 94] considera a abordagem de Conrath para validação técnica e social de sistemas especialistas, embora considere as características de adequação da tarefa para sistemas especialistas e re-elaboração do trabalho como aspectos sociais. Quanto à verificação de sistemas especialistas, este autor considera aspectos comportamentais e ontológicos desses sistemas. Os aspectos comportamentais, referem-se ao desempenho externo esperado quando o sistema estiver em operação, devendo, portanto, ser verificado o que o sistema pode fazer (seu escopo e alcance), como é o desempenho do sistema (sua correção), como é o seu comportamento (sua robustez, transparência, facilidade de aprendizado e operação, eficiência de tempo e recursos) e a confiabilidade do sistema. Os aspectos ontológicos referem-se à estrutura interna (organização e componentes) que produzem o comportamento observável, devendo ser verificada a estrutura do projeto (como a estrutura de representação do conhecimento e os algoritmos de raciocínio) no que diz respeito, por exemplo, à adequação e especificidade; os componentes do software no que diz respeito à correção, manutenção, facilidade de testes, reutilização, portatilidade e compatibilidade com outros sistemas; e finalmente, o conteúdo da base de conhecimento no que diz respeito à consistência, concisão, validade, completude e manutenção [Guida 93].

A verificação da base de conhecimento é considerada de grande importância na literatura, sendo abordada por vários autores [Suwa 82], [Nguyen 87], [Cragun 87], [Ginsberg 88], [Botten 89], [Polat 93], [Jafar 93], [Harrison 94], [Zhang 94]. As

características mais discutidas neste sentido, consistem na verificação da consistência e completude da base.

Verificar a consistência implica na análise das discrepâncias, ambiguidades e redundâncias no conjunto de regras da base de conhecimento [Cragun 87]. Para garantir a consistência da base de conhecimento devem ser verificadas a ausência de regras redundantes, conflitantes, incluídas, circulares, condições desnecessárias e valores de atributos ilegais [Suwa 82], [Naser 88], [Nguyen 87], [Merlevede 91], [Polat 93], [Zhang 94]. Devido à importância desses aspectos, vamos definir cada uma das características identificadas, e utilizaremos a notação do cálculo de predicados para as definições formais, sendo x, y e z as variáveis e p, q e r as relações lógicas:

- *regras redundantes*: duas regras são redundantes se elas ocorrem na mesma situação e têm a mesma conclusão; formalmente na notação do cálculo de predicados dizemos que a regra $p(x) \rightarrow q(x)$ é equivalente a regra $p(y) \rightarrow q(y)$;

- *regras conflitantes*: duas regras são conflitantes se elas ocorrem na mesma situação mas tem conclusões conflitantes ou contraditórias; formalmente na notação do cálculo de predicados dizemos que a regra $p(x) \rightarrow not(q(x))$ é contraditória a regra $p(x) \rightarrow q(x)$;

- *regras incluídas*: uma regra é incluída em outra se as duas regras têm a mesma conclusão mas uma contém condições adicionais. Quando a regra mais restritiva ocorre a menos restritiva também ocorre, o que implica em uma redundância [Suwa 82]; formalmente na notação do cálculo de predicados dizemos que a regra $(p(x)\ and\ q(y)) \rightarrow r(z)$ é incluída pela regra a $p(x) \rightarrow r(z)$;

- *condições desnecessárias*: duas regras contém condições (*if*) desnecessárias se elas tem a mesma conclusão, e uma condição em uma regra está em conflito com uma condição na outra regra, sendo que todas as outras condições nas duas regras são equivalentes; formalmente na notação do cálculo de predicados dizemos que se tivermos as regras $(p(x)\ and\ q(y)) \rightarrow r(z)$ e $(p(x)\ and\ not(q(y))) \rightarrow r(z)$ a condição envolvendo $q(y)$ em cada regra é desnecessária;

- *regras circulares*: um conjunto de regras são circulares se o encadeamento destas regras formam um ciclo, formalmente se apresentam na notação do cálculo de predicados quando temos um conjunto $p(x) \rightarrow q(x)$, $q(x) \rightarrow r(x)$ e $r(x) \rightarrow p(x)$ ou uma regra isolada do tipo $p(x) \rightarrow p(x)$;

- *valores de atributos ilegais*: ocorre em regras que referem-se a valores de atributos que não estão no conjunto de valores possíveis para aquele atributo. Nguyen [Nguyen 87] considera esta característica para verificação da completude;

Completude significa que a base de conhecimento está preparada para responder a todos as possíveis situações que podem surgir dentro do seu domínio do problema [Nguyen 87]. Para garantir a completude da base de conhecimento devem ser cada a ausência de atributos não referenciados, conclusões não alcançáveis, *dead-end* e porções do domínio não cobertas (ausência de conhecimento) [Suwa 82], [Naser 88], [Nguyen 87], [Merlevede 91], [Polat 93], [Zhang 94]. Definiremos, a seguir, cada uma dessas características:

- *atributos não referenciados*: um valor de atributo não referenciado ocorre, quando algum valor de um conjunto possível de valores de um objeto, não é utilizado por nenhuma condição de qualquer regra;

- *conclusões não alcançáveis*: conclusões que nunca são deduzidas, nem podem ser questionadas [Merlevede 91], isto é, toda conclusão de uma regra ou é a conclusão (objetivo do sistema) ou deve ser uma premissa de uma outra regra, se não existe nenhuma dessas situações para uma conclusão, então ela é dita não alcançável [Nguyen 87];

- *dead-ends*: dizemos que existe um objetivo "morto" (*dead-end*) quando os atributos necessários para determiná-lo não são nem questionados ao usuário nem obtidos a partir de uma regra do conjunto de regras;

- *porções do domínio não cobertas*: refere-se à ausência de algum elemento essencial do domínio do problema na base de conhecimento [Merlevede 91], ou seja, existe uma situação na qual um resultado específico é requisitado, mas nenhuma regra da base de conhecimento pode ser ativada nesta situação de forma que produza o resultado desejado [Suwa 82].

3.4.3 Verificação de Sistemas Especialistas

De forma geral, o desenvolvimento de software gera produtos intermediários entre os requisitos iniciais do projeto e o produto final. Estes produtos vão de representações abstratas a representações mais concretas (código final). O processo de verificação envolve a garantia de que cada produto gerado ao longo do desenvolvimento é a

expressão correta do que se deseja, em termos da implementação de todos os requisitos e está sendo produzido corretamente. [Schultz 88]

Muito se tem discutido sobre a problemática de verificação. Morell [Morell 88] sintetiza a abordagem de O'Keefe [O'Keefe 87] definindo as questões referentes a: *o que verificar*, *contra o que verificar* e *com o que verificar*.

Dois componentes definem o primeiro questionamento em relação a *o que verificar*: a base de conhecimento e a máquina de inferência. Atualmente, a grande maioria dos sistemas especialistas são construídos utilizando-se ferramentas (*shells*) que já oferecem o mecanismo de inferência, não sendo necessário portanto verificá-lo. Mecanismos de inferência construídos de forma customizada para o sistema que se está desenvolvendo podem conter falhas e efeitos indetermináveis no processamento, mesmo que a base de conhecimento seja perfeita. No entanto, a base de conhecimento é o componente crucial para a verificação de sistemas especialistas. [Morell 88]

Em relação a *contra o que verificar*, Morell afirma que verificação pressupõe a existência de uma especificação. É certo que uma especificação completa do sistema muitas vezes não é possível, mas, geralmente, existem muitas propriedades desejáveis necessárias que podem ser usadas como especificações parciais ou incompletas. Morell considera este conhecimento sobre a base de conhecimento (meta-conhecimento) de vital necessidade para a verificação.

O último questionamento sobre quais ferramentas e técnicas podem ser usadas para verificação da base de conhecimento, não possui uma resposta única e bem definida. Muitas ferramentas tem sido desenvolvidas para aplicações de sistemas especialistas específicos que utilizam linguagens específicas. Quanto à justificativa da construção dessas ferramentas que simplesmente ajudam na depuração/verificação das bases de conhecimento, Suwa [Suwa 82] diz que elas podem rapidamente identificar problemas da base de conhecimento e possivelmente permitir aos especialistas descobrirem falhas nos seus conhecimentos ou erros em seus raciocínios.

A literatura apresenta várias abordagens para verificação de bases de conhecimento. Muitos autores propõem ferramentas para estas abordagens e aplicam suas idéias em projetos reais. Botten [Botten 89] fez uma coletânea das abordagens e projetos nos quais foram aplicados as verificações e identificou as seguintes abordagens: meta-conhecimento, tabelas, tabelas de decisão, dedução, valores de confiança, matrizes, grafos, redes, processo hierárquico analítico, verificação cedo no ciclo de vida, verificação através de execução. A seguir, vamos descrever algumas dessas abordagens pela apresentação de ferramentas que as utilizam.

3.4.3.1 Ferramentas para Verificação de Sistemas Especialistas

O programa TEIRESIAS, em 1976, desenvolvido com o objetivo de realizar aquisição automática do conhecimento, foi a primeira tentativa de utilizar uma abordagem (tabelas) para verificação automatizada [Polat 93]. TEIRESIAS, utilizado no contexto do trabalho do MYCIN, examinava o conjunto de regras do sistema e construía um modelo mostrando um número de fatores, incluindo quais argumentos eram utilizados para concluir outros argumentos. Dessa forma, analisava a consistência e completude da base de conhecimento [Nguyen 87].

Em 1982, o ONCOCIN também utilizou um programa para verificação de regras (*Rule Checker Program*), baseado na construção de tabelas, para consistência e completude da base de conhecimento. Este programa era uma versão modificada do TEIRESIAS que pode identificar interações incorretas entre regras, verificar regras quanto à sintaxe e semântica, gerar explicação de mecanismo de raciocínio do sistema e automaticamente comparar o resultado do sistema contra conclusões registradas dos especialistas do domínio. Esta comparação inclui uma explicação de porque o sistema obteve conclusões corretas e porque não obteve. Embora este programa facilite a depuração incremental da base de conhecimento, ele requer que o mecanismo de raciocínio do sistema esteja completamente funcional antes de serem iniciados os testes. [Suwa 82]

Tanto TEIRESIAS quanto o verificador de regras do ONCOCIN são possíveis de serem utilizados à medida em que o sistema está sendo desenvolvido. Ambos examinam um conjunto de regras à medida em que elas são introduzidas no sistema. Os problemas da base de conhecimento são detectados primeiro, dividindo as regras em conjuntos distintos, cada qual concluindo sobre o mesmo parâmetro dentro do mesmo contexto. Os conjuntos de regras resultantes podem ser verificados independentemente. A partir daí, constrói-se uma tabela, mostrando todas as possíveis combinações de valores dos parâmetros utilizados nas condições e os valores correspondentes que irão ser concluídos pela ação do parâmetro. É verificada, então, a consistência e a completude na tabela. Finalmente, uma tabela é exibida com um resumo de todos os potenciais erros encontrados no que se refere a regras conflitantes, redundantes, incluídas e ausência de regras. Os programas assumem que existe uma regra para cada combinação de valores dos parâmetros das condições. Isto implica na hipótese de ausência de regras o que pode resultar em regras hipotéticas que têm semanticamente combinações impossíveis de parâmetros. Além disso, se o número de parâmetros é grande o programa poderá sugerir um grande número de regras [Suwa 82], [Nguyen 87].

CHECK [Nguyen 87], [Polat 93] é uma extensão do programa de verificação de regras utilizado no ONCOCIN, sendo uma ferramenta para verificação da base de conhecimento construída por LES (*Lockheed Expert System Shell*). CHECK difere do trabalho do ONCOCIN por ser aplicado a um conjunto inteiro de regras para um objetivo, e não somente a um subconjunto que determina valores de cada parâmetro. Com essa visão global, CHECK inclui verificação de aspectos ainda não considerados (conclusões não alcançáveis, dead-ends, condições desnecessárias e valores de atributos não referenciados ou ilegais). O programa faz a verificação através da construção de duas tabelas: uma que mostra a interação entre as regras dirigidas a dados (*data-driven*) e outra que mostra a interação entre regras dirigidas à conclusão (*goal-driven*) e aos objetivos.

Cragun [Cragun 87] utilizou tabelas de decisão para verificação da consistência e completude de sistemas especialistas baseado em regras do programa ESC (*Expert System Checker*). A tabela de decisão é caracterizada pela separação entre condições e ações em um sentido e em valores e expressões de condição no outro (Figura 3.2). Toda coluna da tabela (coluna de decisão) indica que ações devem ou não ser executadas para uma combinação específica de estados de condição, de forma que todo caso possível é incluído em uma e somente uma coluna. A abordagem orientada à condição, torna a tabela de decisão muito útil para mostrar o conhecimento oferecendo uma visão geral, facilidade de leitura e facilidade para verificação da completude e consistência [Merlevede 91].

Condição / Regra	Regra 1	Regra 2	....	Regra N
Condição 1	V	V		F
Condição 2	V	F		F
Condição 3		V		F
....				
Ações				
Ação 1	x	x		
Ação 2		x		
Ação 3				x
Ação 4				x
...				

Figura 3.2 - Modelo de tabela de decisão

Na verificação do ESC, primeiramente é construída uma tabela de decisão mestre para toda a base de conhecimento pela leitura de um arquivo de regras (as regras devem estar escritas em linguagem natural em um formato previamente especificado). Esta tabela vai sendo expandida à medida em que novas condições e ações são encontradas, e cada valor correspondente sendo incluído na linha e coluna apropriadas. Depois, a tabela é particionada lógica e automaticamente, em conjuntos de contextos distintos. A partir daí, inicia-se a verificação que consiste de três passos: *(i)* verificar a ambiguidade e redundância, pela comparação de cada regra com as demais regras dentro da mesma sub-tabela; *(ii)* verificar a completude e *(iii)* verificar a ausência de regras. [Cragun 87]

Ginsberg [Ginsberg 89] propôs uma outra abordagem para verificação da base de conhecimento no que se refere a consistência e redundância: a redução da base de conhecimento. Segundo este autor, esta técnica é melhor do que as outras porque, em princípio, ela pode detectar todas as potenciais contradições e redundâncias que existem em bases de conhecimento. O *KB-Reducer* (*Knowledge-based Reducer*) é um sistema que implementa uma versão especial da técnica de redução da base de conhecimento. O *KB-Reducer* é baseado em um modelo abstrato de inferência que enfatiza a relação entre formas de certeza do raciocínio do sistema especialista e dedução natural em lógica proposicional, analisando base de conhecimento escrita numa linguagem de representação de regras canônica. Ginsberg admite que a redução da base de conhecimento é um primeiro passo de um procedimento para prover os sistemas especialistas com a capacidade de automaticamente modificarem sua base de conhecimento para se adaptar a ambientes variados.

UVT (*Unification-based Verification Tool*) é uma outra ferramenta para verificação da base de conhecimento descrita por Polat [Polat 93]. As regras da base de conhecimentos com predicados comuns podem estar relacionadas. Durante a verificação, UVT compara os literais para determinar as relações entre eles. Estes predicados comuns podem ser equivalentes embora não seja exatamente iguais. Os predicados em conjunção podem estar em qualquer ordem, embora poderia ter alguma restrição imposta pela lista de substituição. Para identificar a equivalência de predicados comuns é utilizado a técnica de unificação[2]. O programa é iniciado pela análise de todas as regras inferenciadas e inclusão numa base de regras. O programa considera que as regras estão segundo um formato estabelecido. O segundo passo é construir duas tabelas a partir das relações entre as regras. Uma tabela contém a informação relacional obtida pela comparação dos parâmetros (consequente e antecedente) de pares de regras usando

[2] Unificação é um algoritmo para determinação de substituições necessárias para tornar duas expressões do cálculo de predicados iguais (Luger, G.F. e Stubblefield, W.A.; Artificial Inteligence and the Design of Expert Systems, The Benjamin/Cummings Publishing Company, Inc, pag 62; 1989)

unificação. A segunda tabela, é utilizada para detectar possíveis regras circulares. A partir dessas tabelas o UVT verifica a base de conhecimento para os requisitos de completude e consistência.

Uma abordagem relativamente nova para verificação da base de conhecimento é a utilização de redes Petri, utilizada na ferramenta PREPARE proposta por Zhang [Zhang 94] para verificação de regras inconsistentes, redundantes, incluídas, circulares e incompletas da base de conhecimento. PREPARE é baseado na modelagem da base de conhecimento em tipo especial de redes Petri chamado redes *Predicate/Transition* (Pr/T) que são representações gráficas da lógica de predicados de primeira ordem. No PREPARE, as redes Pr/T são usadas para descrever as relações entre uma regra e as diferentes regras e fatos da base de conhecimento. A partir daí, a verificação da base se faz pela identificação de padrões de estruturas de subredes na rede Pr/T. Para cada característica a ser verificada é definido um padrão de estrutura. O PREPARE contém, portanto, quatro componentes: um *transformador*, que traduz as regras e fatos da base de conhecimento em uma representação de rede Pr/T; um *verificador* que detecta e localiza as fronteiras dos padrões inconsitente, redundante, circular e incompleto; um *formulador* que codifica e formula um texto para cada padrão inconsistente/redundante/ circular; e, um *classificador*, que determina o tipo de padrão pelo reconhecimento do texto produzido pelo formulador e descobre o tipo de incompletude pela análise dos resultados obtidos pelo verificador.

3.4.4 Validação de Sistemas Especialistas

Validação de sistemas especialistas é mais do que apenas um processo voltado para encontrar erros. Validação é o processo de determinação de que um sistema especialista representa o conhecimento de um especialista [O'Leary 90]. Isto significa que o processo de validação implica na investigação do que o sistema conhece, do que não conhece ou conhece de modo incorreto; na análise do nível de especialidade na tomada de decisão do sistema; na determinação de se o sistema especialista é baseado em teoria e na análise de sua confiabilidade [O'Leary 87].

Alguns questionamentos sobre o processo de validação são constantemente abordadas na literatura referindo-se a *o que validar*, *quando validar*, *como validar*, *como controlar o custo da validação*, *como controlar o preconceito contra validação* e *como lidar com múltiplos resultados* [O'Keefe 87], [O'Leary 90], [Mengshoel 93].

O'Keefe [O'Keefe 87] responde a estes questionamentos, afirmando que devemos validar qualquer resultado intermediário, o resultado final, o raciocínio do sistema ou qualquer combinação desses três. Claro que isso implica em validar o processo de raciocínio, porque um processo ruim com resultados corretos não pode ser adequado para um domínio maior. Tipicamente, devemos validar o processo de raciocínio o mais cedo possível e o resultado final quando a base estiver mais completa. Para Harrison [Harrison 94] a validação do processo requer a elaboração do processo dinâmico do raciocínio em termos de algumas características, como modelos de causas, dependências sequenciais entre inferências, tarefas ordenadas ou dependências de objetivos.

O que validar está intrinsecamente relacionado com o estágio de desenvolvimento que está sendo realizado, ou seja, com o momento em que está sendo realizada a validação. Vários autores incluem o processo de validação como uma atividade a mais no processo de desenvolvimento do sistema [Geissman 88], [Naser 88], [O'Keefe 87], [O'Leary 90], [Merlevede 91], [Harrison 94].

Em relação a como validar, O'Keefe [O'Keefe 87] propõe que os sistemas especialistas sejam validados tanto com relação a resultados conhecidos quanto com relação à opinião de especialistas. Além disso, sistemas especialistas devem ser validados utilizando-se vários casos, previamente documentados, representando um trabalho de muitos especialistas sobre um conjunto completo de problemas. Uma validação mais detalhada irá requisitar testes do sistema contra um pequeno número de casos complexos e obscuros. Nestes casos o que importa não é o número de testes e sim o escopo que cobre os testes.

É difícil determinar o esforço e custo necessários para realizar a validação de um sistema. O custo deve ser controlado elaborando métodos de validações formais que são integrados dentro do processo de desenvolvimento. O valor da validação, entretanto, está diretamente relacionado com o valor do sistema para seus usuários e o risco envolvido no uso de um sistema não confiável.

Os sistemas especialistas são, algumas vezes, validados com especialistas que sentem sua especialidade ameaçada quando comparada com outros especialistas ou com o próprio sistema. Com isso há preconceito, ou prevenção contra a atividade de validação dos sistemas, dificultando esta atividade [Messeguer 92]. Este problema é solucionado utilizando-se técnicas de validação que omitam a identidade dos avaliadores e, que utilizem na validação especialistas que não tenham participado do desenvolvimento do sistema [O'Keefe 87].

Quando validamos um sistema especialista com múltiplos resultados, existe a dificuldade referente à de um problema com múltiplas respostas (por exemplo em sistemas de prescrição médica para tratamento de uma doença, dois tipos de remédios são válidos se considerados separadamente, mas sua combinação não é aceitável. Portanto, o conjunto de respostas do sistema é inválido). Nestas situações não podemos testar a validade das várias respostas do sistema especialista pela validação de cada resposta separadamente. É necessária uma abordagem múltipla, para incorporar a correlação entre os vários resultados para validar o sistema como um todo.

O'Keefe [O'Keefe 87] considera que uma validação pode ser formal ou informal. Uma validação formal estabelece quando a validação deve ocorrer dentro do ciclo de desenvolvimento e identifica métodos de validação, entrada da especificação do domínio, e, quando apropriado, a relevância de técnicas estatísticas. A validação informal, tipicamente realizada no final do desenvolvimento, emprega métodos *ad-hoc* e é frequentemente de análise subjetiva.

Segundo O'Leary [O'Leary 87] a validação difere em cada situação na formalidade e em até que ponto a validação é implementada.

Existe na literatura técnica, um conjunto de propostas para validar um sistema especialista qualitativa e quantitativamente [O'Keefe 87], [O'Leary 90], que decrevemos a seguir.

3.4.4.1 Validação Qualitativa

Validação qualitativa emprega comparação subjetiva, o que não significa que sejam validações informais. Encontramos na literatura os seguintes exemplos de validações qualitativas: validação face a face, validação preditiva, teste em campo, validação por subsistemas, análise de sensibilidade, interação visual e teste de Turing [O'Keefe 87].

A proposta da *validação face a face* consiste em se avaliar a consistência entre a visão dos desenvolvedores, e a visão dos especialistas, para comparação do desempenho do sistema com relação ao desempenho do especialista humano. Nesta abordagem, podem participar, também, outras pessoas com conhecimento do domínio da aplicação [O'Keefe 87]. Segundo O'Leary [O'Leary 90], esta técnica funciona como um mecanismo de *feedback* para o desenvolvedor refinar a base de conhecimento, reprojetar e reformular fases do processo de desenvolvimento, sendo finalizada quando desenvolvedores, especialistas e outros avaliadores concordarem que o protótipo avaliado reflete adequadamente o problema e está bem estruturado.

Validação preditiva requer a existência de casos históricos para teste e resultados conhecidos ou obtidos de especialistas humanos. A validação ocorre com a comparação dos resultados obtidos do sistema com os resultados conhecidos para os casos históricos de testes. Esta comparação comprova se o sistema possui uma exatidão satisfatória de acordo com o desejado.[O'Keefe 87]

O *teste em campo* é a validação do sistema no ambiente em que será operado. Do ponto de vista dos desenvolvedores, este tipo de teste oferece duas vantagens. Em primeiro lugar transfere para os usuários do sistema a tarefa de realizar os testes. Em segundo lugar, a aceitação do desempenho do sistema é obtida, implicitamente, quando os usuários não tiverem mais queixas com relação ao sistema para relatar. Este tipo de validação só é possível em aplicações não-críticas, onde os usuários podem avaliar a correção do sistema especialista sem riscos. [O'Keefe 87]

Validação por subsistemas requer que o sistema especialista seja decomposto em subsistemas para serem validados [O'Keefe 87]. Nesta abordagem, os subsistemas são validados à medida em que são desenvolvidos. Este tipo de validação possui três vantagens: incorpora a atividade de validação dentro do ciclo de desenvolvimento, são mais fáceis de realizar pois validam subsistemas que são menos complexos e mais gerenciáveis do que um sistema completo e facilita a detecção de erros. Entretanto, pode não ser possível observar o comportamento de entrada/saída de um subsistema e a validação com sucesso de cada subsistema não implica na validade de todo o sistema pois tolerâncias de pequenos erros acumulados podem ser significativas no desempenho do sistema final. Dessa forma, como na validação face a face, o enfoque desta abordagem é nos detalhes do protótipo que está sendo avaliado e serve para identificar áreas no protótipo que requerem um desenvolvimento ou revisão mais detalhada [O'Leary 90].

A abordagem chamada de *análise de sensibilidade* consiste na alteração dos valores de variáveis e parâmetros de entrada de acordo com alguma variação de interesse para observar os efeitos produzidos pelo sistema. Ou seja, consideremos um sistema que produz um resultado satisfatório para um caso de teste X que utiliza n entradas. A análise de sensibilidade consiste na alteração de cada uma dessas n entradas e análise do efeito sobre o resultado oferecido pelo sistema. Este tipo de validação é muito útil para sistemas que trabalham com incerteza, já que podem ser alterados quando desejado e o efeito das medidas de incerteza intermediários e final pode ser examinado. [O'Keefe 87]

A validação através de *interação visual* provê animação visual do sistema especialista trabalhando, e permite ao especialista interagir alterando parâmetros quando desejado. A interação visual pode ser vista como um ambiente para validação face a face

interativa, validação por subsistemas interativa e análise de sensibilidade interativa. Este tipo de validação tem sido muito utilizado na validação de modelos de pesquisa operacional.[O'Keefe 87]

O *teste de Turing* [Chandrasekaran 83], [O'Keefe 87], [O'Leary 90], [Harrison 94] consiste na validação do sistema especialista comparando o seu desempenho com o de especialistas humanos. Esta avaliação é realizada por especialistas sem que estes tenham conhecimento de quem (sistema ou os especialistas) está sendo avaliado. Chandrasekaran [Chandrasekaran 83] propõe que os especialistas que participam do teste de Turing obrigatoriamente não tenham participado do desenvolvimento do sistema. Inicialmente, um grupo de especialistas (que chamaremos de G1) gera um conjunto de casos de testes (casos gerais de sua experiência como especialista e casos interessantes que procurem fazer com que o sistema falhe). Estes casos, são submetidos a outro grupo de especialistas (G2) e ao próprio sistema especialista para geração de resultados. O grupo inicial (G1) organiza os resultados obtidos suprimindo a identidade dos diagnosticadores (especialistas humanos e sistema especialista), tendo portanto um papel neutro no processo de avaliação. Chandrasekaran sugere organizar os resultados em duas estruturas: uma apenas com os resultados finais para cada caso, e outra colocando todas as informações consideradas e rejeitadas no diagnóstico. Finalmente, os casos e as conclusões são submetidas a um grupo de especialistas que deve atribuir um grau de desempenho às respostas geradas, desconhecendo quais delas foram geradas pelo especialista humano e quais pelo sistema especialista. O'Leary [O'Leary 90] sugere que este grupo seja diferente dos dois grupos que participaram do processo nas primeiras fases, enquanto que Chandrasekaran [Chandrasekaran 83] sugere que seja o próprio grupo G2. Chandrasekaran sugere, ainda, com sua abordagem de colocar os resultados em duas estruturas, que primeiramente seja submetida à avaliação a estrutura apenas com os resultados finais para que a precisão do diagnóstico seja feita sem a base dos elementos percebidos pelos diagnosticadores expressos na outra estrutura mais completa. Esta técnica foi utilizada para validar os sistemas especialistas MYCIN e ONCOCIN.

3.4.4.2 Validação Quantitativa

A validação quantitativa emprega técnicas estatísticas para comparar o desempenho do sistema especialista com casos de testes ou especialistas humanos.

Os métodos de validação quantitativa seguem duas categorias. Na primeira é definido um intervalo de confiança para uma ou mais medidas que são comparadas

subjetivamente com uma variação de desempenho aceitável. A segunda utiliza um teste de hipótese para comparar medidas com um valor predeterminado de desempenho aceitável, determinando se o sistema é valido ou não. [O'Keefe 87]

Existem muitos métodos quantitativos na literatura. O'Keefe [O'Keefe 87] discute três deles: o *paired t-test*, o *Hotelling's one sample* T^2 *test* e intervalos de confiança simultâneos. O *paired t-test* é adequado para comparar diferenças entre resultados obtidos, podendo ser utilizado para medir diferenças entre o sistema especialista e os especialistas humanos. O *Hotelling's one sample* T^2 *test* é adequado para validação de sistemas que produzem múltiplos resultados, provendo correlação entre eles. Intervalos de confiança simultâneos são, também, utilizados para sistemas especialistas com múltiplas respostas, através da definição de intervalos de confiança ou junção de regiões de confiança para diferenças de pares de respostas.

3.4.4.3 Ferramentas para Validação de Sistemas Especialistas

A validação de sistemas especialistas tem sido amplamente abordada, no sentido de procurar metodologias e ferramentas que reduzam o esforço necessário para validação desses sistemas. A seguir, vamos relatar algumas ferramentas apresentadas na literatura.

A utilização de prototipagem no desenvolvimento de sistemas especialistas tem alcançado sucesso. O processo de validação sobre os protótipos de demonstração implica em um melhor detalhamento do problema e garantia de um desenvolvimento do sistema especialista desejado. Além disso, é a sucessiva validação de protótipos que possibilita uma base de conhecimento completa.

Muitos autores valorizam o protótipo inicial ou de demonstração. Geissman [Geissman 88] na metodologia para verificação e validação que propõe, considera o desenvolvimento inicial de um protótipo como a base para estabelecer os requisitos do sistema, sem os quais não é possível realizar qualquer avaliação. Mesmo sabendo que estes protótipos podem ser incompletos ou que, aos poucos, vão se perdendo, ele considera que a sua construção dá uma clara idéia do problema a partir da perspectiva do engenheiro do conhecimento e que esta idéia permite a preparação para validação sobre o sistema especialista em relação ao que ele irá exatamente fazer.

O'Leary [O'Leary 90] coloca os protótipos como um componente de interação que relaciona processos de avaliação (Figura 3.3). A partir desta interação entre especialistas, protótipos e engenheiros do conhecimento (desenvolvedores e avaliadores) durante o processo de validação, à medida que os participantes encontram inconsistências ou

limitações não aceitáveis no protótipo, fazem reformulações, reprojetam, refinam e corrigem as tarefas. Dessa maneira, a validação se torna a condutora da evolução de um protótipo inicial para o sistema em si.

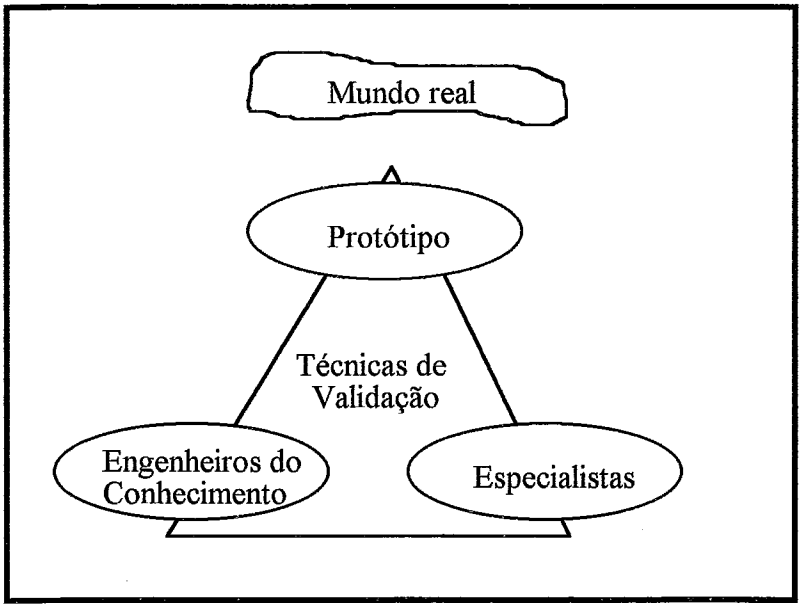

Figura 3.3 - Interação no paradigma de validação de O'Leary [O'Leary 90]

Segundo Geissman [Geissman 88] a descrição de um protótipo é a base ideal para os requisitos, porque o protótipo é um programa de computador real e avaliável, sendo que cada protótipo gerado provê um conjunto parcial de requisitos para o próximo protótipo a ser construído.

Krishnamurthy [Krishnamurthy 87] descreveu um método e uma ferramenta desenvolvida para teste e validação de bases de conhecimento (*Test and Validation Environment*). O método baseia-se nas seguintes premissas: bases de conhecimento tem um caráter declarativo e representam informações estruturais sobre algum domínio de aplicação; as condições de integridade, consistência e correção podem ser uniformemente representadas por grafos e verificadas por algoritmos sobre grafos. O ambiente interativo de teste e validação (TVE) é independente do ambiente que especifica a base de conhecimento para que possa ser adaptado para diferentes problemas e linguagens de representação. O TVE recebe a base de conhecimento a ser testada que é mapeada em grafos através de um componente chamado mapeador (*mapper*). A partir daí, executa um conjunto de algoritmos para grafos, gerando a saída de erros e problemas para o usuário. Este processo de análise é interativo e suportado

por gráficos. A partir desta experiência, os autores perceberam que o resultado da avaliação depende da abordagem de programação de inteligência artificial utilizada e que o caráter declarativo das base de conhecimento permite a aplicação de técnicas automáticas para validação e testes que podem melhorar significativamente a segurança e a confiabilidade de grandes softwares.

A partir da observação de que as bases de conhecimento, que representam o conhecimento adquirido de especialistas, devem ser validadas com relação à própria fonte de conhecimento, Mengshoel [Mengshoel 93] implementou uma ferramenta para validação do conhecimento através de casos. O KVAT (*Knowledge Validation Tool*) valida objetos do conhecimento representados em *frames* ou proposições, usando casos definidos pelos especialistas. A ferramenta provê um conjunto de funções que permite inspecionar, registrar e excluir casos de validação com descrições, comparar a solução da base de conhecimento com as soluções dos especialistas e incluir ou excluir proposições da base de conhecimento através de um editor.

A ferramenta EVA (*Expert System Validation Associate*) [Stachowitz 87] foi desenvolvida com o objetivo de melhorar o processo de validação pela procura de erros e omissões na base de conhecimento. EVA foi escrita em LISP e ART (*Automated Reasoning Tool*) para sistemas especialistas baseados em ART e compreende, na verdade, verificação e validação de sistemas especialistas combinando análise estática e dinâmica. EVA interage com a ferramenta de sistemas especialistas através de uma interface específica. Então, algoritmos de conversão traduzem regras, fatos e esquema de um objeto da ferramenta no formato da ferramenta EVA. As regras, fatos e esquema de uma aplicação de sistemas especialistas são verificadas quanto a erros estruturais, lógicos e semânticos. Tanto a verificação estrutural quanto a lógica consistem de dois passos. O primeiro é desempenhado por meio de algoritmos especiais sem a informação semântica. No segundo, chamado de verificação estendida, utiliza a informação semântica contida numa base de meta-conhecimentos. A verificação semântica é desempenhada através de meta-regras que operam sobre outras meta-regras, meta-fatos, regras e fatos de objetos. EVA aumenta a confiabilidade do sistema especialistas, acelera seu desenvolvimento e ajuda no processo de contínua modificação.

Franklin [Franklin 88] desenvolveu um sistema, o ESPE (*Expert System Parsing Enviroment*), que ajuda na depuração e teste de sistemas especialistas pelo rastreamento do fluxo de informações e ajuda na manutenção pela realização de análise de sensibilidade. Os sistemas especialistas a serem analisados são escritos em ESDE (Expert System Development Enviroment) da IBM. O sistema especialista é convertido em grafos acíclicos onde um par de valores de parâmetros ou uma regra é um nodo e uma

instância de regra para pares de valores de parâmetros são arcos entre as regras e os valores de parâmetros. Sobre este grafo, o sistema conta o número de caminhos entre todos os pares de nodos e gera uma tabela para análise, é realizada análise de sensibilidade para os valores que os parâmetros podem assumir. Com isto, os autores acreditam que a ferramenta permite ao usuário testar a completude do sistema especialista para o problema, dá ao usuário um sentimento intuitivo de como a informação está fluindo no sistema e ajuda o usuário a julgar a adequação de sistemas especialistas para solução da tarefa.

Um conjunto de sete ferramentas de validação foi desenvolvido no projeto VALID [Meseguer 92] do Esprit-II para apoiar engenheiros do conhecimento numa variedade de tarefas de validação exigidas na construção de um sistema baseado em conhecimento como um todo. O ambiente do VALID é um software que integra este conjunto de ferramentas e oferece uma interface gráfica para dar suporte ao processo de validação. Uma das ferramentas é o CONCRET, utilizada para suporte ao refinamento do conhecimento e revisão do conhecimento estratégico da base de conhecimento através do processamento automático de um conjunto de casos exemplo, análise do processo de raciocínio dos casos errados e exibição para o engenheiro do conhecimento dos possíveis erros da base de conhecimento. O CONTROLLER também usa um conjunto de casos para verificar a coerência dos dados de entrada solicitados, do resultado gerado e do resultado com relação às entradas. CURRICULUM KB é usado durante a implementação e teste da base de conhecimento para inspecionar sua evolução através de seus diferentes objetos (regras, instâncias, etc.). Para acompanhar a execução existe o EV (EXECUTION-VISUALIZER), que oferece um rastreamento da execução para um caso de entrada, permitindo, ainda, se colocar pontos de controle com informações específicas. A consistência da base de conhecimento é verificada pela ferramenta chamada IN-DEPTH a partir de restrições de integridade estabelecidas, incluindo também gerenciamento de incerteza e meta-controle. Outra ferramenta para verificação da consistência da base de conhecimento é o PENIC; entretanto este, somente, é utilizado para sistemas baseado em lógica proposicional, sem meta-controles ou incerteza. A última ferramenta é a WITNESS que serve para registrar as opiniões de diferentes pessoas (usuários finais, especialistas e engenheiros do conhecimento) sobre o sistema baseado em conhecimento que está sendo avaliado. Dessa forma, é criada uma *base de comentários* que pode ser referenciada pela equipe de desenvolvimento para propor modificações bem fundamentadas.

3.5 Metodologias de Verificação e Validação

Segundo Green [Green 87] a falta de uma metodologia para verificação e validação de sistemas especialistas implica na necessidade de tentativas de diferentes abordagens para seleção de métodos apropriados que serão mais aplicáveis e fornecerão melhores resultados em cada circunstância. Realmente, o que se encontra na literatura são experiências práticas em diferentes áreas de aplicação e com generalização limitada [Guida 94].

Segundo Guida [Guida 93], [Guida 94] as abordagens de verificação e validação de sistemas especialistas são divididas em três classes: *(i)* abordagens orientadas à base de conhecimento, que se preocupam com possíveis problemas lógicos do conhecimento (como consistência, completude, redundância, etc.), *(ii)* abordagens baseadas em critérios de avaliação, que se preocupam com o que uma avaliação deve medir e propõem critérios para verificação e validação, e, *(iii)* abordagens baseadas em métodos de avaliação, que se preocupam em como o sistema especialista deve ser avaliado, sendo, muitas vezes, encontradas como parte do ciclo de vida destes sistemas, incluindo avaliações qualitativas e quantitativas.

A seguir, vamos descrever algumas abordagens apresentadas na literatura, dando um maior destaque a proposta de cada autor para verificação e validação de sistemas especialistas.

A primeira abordagem metodológica que iremos tratar é a proposta por Green em 1987 [Green 87]. Sua proposta nada mais é do que a inclusão de atividades específicas de sistemas especialistas no processo de desenvolvimento de um sistema, compreendendo cinco etapas: definição dos requisitos, verificação da base de conhecimento e software de suporte, preparação de casos de teste, execução dos testes e avaliação dos resultados. A etapa de definição de requisitos é fundamental para o processo de verificação e validação de sistemas especialistas podendo exigir muitas interações entre usuários e desenvolvedores pois os requisitos são, às vezes, difíceis de serem determinados. Especificação de requisitos para sistemas especialistas possui diferenças em relação à especificação de requisitos dos sistemas convencionais (como discutido na seção 3). A partir da especificação de requisitos é realizada a verificação. A seleção de casos de teste deve ser cautelosa procurando testar todos os requisitos, todas as possíveis decisões a serem geradas e testar não só a saída do sistema mas o processo pelo qual o sistema determinou a saída. A partir da execução dos casos de teste é realizada a avaliação. Green considera que a avaliação deve ser realizada por especialistas que não participaram do desenvolvimento para tornar o processo mais independente. Esta avaliação deve levar em conta se o resultado esperado ou um

resultado aceitável foi obtido, se a justificativa para os resultados (no caso de sistemas que geram a justificativa) está na forma adequada para o usuário, se o processo de inferência utilizado pelo sistema está suficientemente completo e robusto, se o conhecimento empregado é suficientemente profundo para garantir ao usuário confiança no sistema e caso não for esperado que o sistema alcance cem por cento de acerto deve, também, ser analisado o impacto dos resultados incorretos.

A abordagem de Geissman [Geissman 88] considera um processo de verificação e validação formal para sistemas especialistas em áreas críticas. A metodologia que propõe tem aspectos do processo de desenvolvimento clássico, especialmente na decomposição *top-down*, e é formada por seis etapas. A primeira etapa consiste na definição de requisitos a partir do desenvolvimento de um protótipo rápido para melhor entendimento do problema a ser resolvido. A segunda etapa consiste da elaboração do projeto em termos de paradigmas formais (como frames ou regras de produção) para garantir uma implementação analisável, isso implica na escolha de um ou mais paradigmas de processamento do conhecimento que será utilizado. Com isso, é realizada a terceira etapa que é a certificação de que a máquina de inferência suporta todos os paradigmas escolhidos. Esta etapa envolve: definição formal de um paradigma de inferência, especificação de como este paradigma é representado na ferramenta utilizada pelo sistema, desenvolvimento de testes adequados para o paradigma numa representação abstrata, tradução destes testes para a linguagem da ferramenta específica e a certificação dos testes. A quarta etapa refere-se à verificação a nível de um projeto de alto-nível, consistindo da determinação de que o projeto atende a todos os requisitos. Depois que a verificação do projeto de alto-nível estiver completo, o desenvolvimento e verificação passa para um nível mais baixo, com mais detalhes ou diretamente no código, realizando, assim, a verificação da base de conhecimento. Esta verificação deve ser realizada por outras pessoas que não sejam os desenvolvedores. Deve ser confirmado que a base de conhecimento ajusta-se com o paradigma especificado, verificar a estrutura dos subproblemas determinados, confirmar que a base de conhecimento está correta ou razoável por uma análise estática do código e, possivelmente, incluir regras para assinalar falhas de certas condições críticas. A última etapa consiste da validação formal do desempenho com o teste do comportamento do sistema. Os autores propõem a utilização de prova formal. Para a validação é necessário identificar os critérios de qualidade que se deseja alcançar, definir métricas objetivas para estes critérios, desenvolver uma biblioteca de casos de testes, validar o sistema com especialistas não envolvidos no projeto, testar o sistema em paralelo com os métodos não-automatizados por um período de tempo comparando os resultados e manter informações detalhadas do desempenho do sistema à medida que a base de conhecimento é elaborada.

A abordagem de O'Leary [O'Leary 87] difere das anteriores por não se tratar de uma metodologia de desenvolvimento com preocupação com a validação e verificação de sistemas especialistas, e sim uma organização do trabalho para validação desses sistemas. A Figura 3.4 mostra o sumário do que é observado. O primeiro aspecto refere-se à validação do conteúdo através do exame direto da base de conhecimento pelos especialistas, do teste de Turing e da avaliação do sistema com relação a outros modelos para o mesmo problema. Para a validade de critérios, O'Leary discute a dificuldade em estabelecer medidas para definir o nível de especialidade desejada, determina que deve ser avaliada a consistência, completude e adequação da base de conhecimento e que o método para avaliar estes critérios é particular para cada aplicação. A validade da construção refere-se à importância da existência de uma teoria na qual o sistema é baseado, pois a utilização de uma abordagem empírica simplesmente não é tão eficiente quando se desenvolve um sistema especialista quanto uma abordagem baseada em uma teoria . A objetividade refere-se a minimizar as variações no julgamento do sistema com a eliminação da prevenção do especialista ao sistema pela administração de testes independentes, o uso de técnicas que omitam a identidade dos participantes na validação e com a realização de testes informais periódicos realizados pelo programador. A análise do custo-benefício é fundamental na validação, apesar de ser uma tarefa difícil de ser realizada. Para analisar os benefícios deve ser analisado como o sistema será utilizado (comercialmente, benefícios sociais, etc) e quanto ao custo para validar um sistema deve ser analisada sua formalidade e até que ponto a validação é requerida. Para garantir a confiabilidade do sistema deve ser realizada análise de sensibilidade e padrões de problemas de testes utilizados para revalidação do sistema para garantir, por exemplo, que a adição de conhecimentos na base de conhecimento não resulta em contradições. O'Leary sugere que deve existir uma variação sistemática nos testes para o sucesso da validação. Portanto, deve-se escolher testes que reflitam uma grande quantidade de problemas encontrados, realizar variações nesses testes, escolher uma boa quantidade de testes. O controle de variação externa significa que a influência de variáveis independentes externas são minimizadas, anuladas ou isoladas. Para isso são considerados fatores como a complexidade do sistema e o aprendizado durante o processo de validação. Finalmente é considerado que deve haver uma minimização da variação de erros geralmente causada por respostas inadequadas do especialista.

1. Validade do Conteúdo
   - Exame direto do sistema por especialistas
   - Teste do sistema contra especialistas humanos (Teste de Turing)
   - Teste do sistemas contra outros modelos

2. Validade de Critérios
   - Definição do nível de validade do sistema
   - Critérios para base de conhecimento
   - Determinação da validação dos critérios

3. Validade de Construção

4. Objetividade
   - Validação pelo programador
   - Administração independente para validação
   - Validação pelo usuário final
   - Técnicas para uma validação que não identifique os avaliadores

5. Custo-benefício

6. Confiabilidade
   - Teste do sistema contra ele mesmo (Análise de sensibilidade)
   - Problemas de teste padrão para revalidação

7. Variação sistemática

8. Variação externa

9. Variação de erro

Figura 3.4 - Organização do trabalho de validação [O'Leary 87]

Guida [Guida 93] propôs uma metodologia para avaliação do desempenho e qualidade de sistemas especialistas, através da medição de cada atributo de qualidade e combinação dos valores obtidos. O autor considera os atributos de acordo com aspectos comportamentais e ontológicos, como descrito na seção 3.4.2., definindo uma relação entre eles. A combinação é definida através de uma função que considera as medidas dos atributos. Estas medidas, geralmente, são valores reais e num intervalo entre 0 e 1 para os atributos comportamentais e um conjunto finito de valores qualitativos (como muito alto, alto, médio, baixo e muito baixo) para os atributos ontológicos. Para obter estas medidas, são utilizados testes com análise de entrada/saída para os atributos comportamentais e inspeções para avaliar os atributos ontológicos. Além disso, o autor sugere que deve ser determinado um peso para cada atributo avaliado de forma que este represente sua importância relativa aos outros na determinação do valor da função. A

função de combinação considera, então, todas estas medidas através de uma média ponderada.

3.6 Conclusão

Neste capítulo abordamos questões relativas à verificação e validação de sistemas especialistas. A partir deste estudo, foi possível concluir que apesar de terem acontecido avanços significativos nesta área, ainda existe muito trabalho a ser realizado. Percebemos na literatura que muitas experiências estão acontecendo em diferentes projetos no que diz respeito à avaliação desses sistemas, principalmente relacionada a base de conhecimento, mas sentimos a necessidade de uma conceitualização maior, mais abrangente e concisa em relação às características de qualidade que devem estar presentes tanto na base de conhecimento como no processo de desenvolvimento desses sistemas (seus produtos intermediários e final). Este tema será abodado no próximo capítulo.

Capítulo 4
Características de Qualidade de Sistemas Especialistas

4.1 Introdução

Com a importância dada a procedimentos para garantia da qualidade de software, vários trabalhos tem sido realizados para definição de atributos de qualidade para os produtos de software, em geral, e considerando as diferentes áreas de aplicação.

Neste capítulo definimos um conjunto de atributos de qualidade que devem ser considerados para a avaliação de sistemas especialistas, tanto no que se refere a especificações do sistema quanto ao produto final. Estes atributos foram organizados de acordo com o modelo proposto por Rocha [Rocha 83] e definidos a partir da ISO 9126 [ISO/IEC 9126], de trabalhos anteriores realizados na COPPE em avaliação da qualidade [Clunie 87], [Palermo 89], [Belchior 92a], [Campos 94a], [Campos 94b] e da literatura técnica na área de sistemas baseados em conhecimento/ sistemas especialistas [Suwa 82], [Waterman 86], [Cragun 87], [Marcot 87], [Green 87], [Nguyen 87], [Stachowitz 87], [Naser 88], [Rushby 88], [Schultz 88], [Bundy 87], [Krishnamurthy 87], [Conrath 91], [Merlevede 91], [Mengshoel 93], [Guida 93], [Guida 94], [Zhang 94].

4.2 Atributos para Avaliação da Qualidade

Produtos de software são desenvolvidos para atenderem a determinadas necessidades de seus usuários. Após serem colocados em operação, espera-se que tenham uma vida útil, longa e produtiva. Para que isto se concretize, devem ser atingidos os seguintes objetivos de qualidade [Rocha 87]: *utilizabilidade*, *confiabilidade conceitual* e *confiabilidade da representação* (Figura 4.1).

Utilizabilidade é um objetivo fundamental, e refere-se às características de utilização do software, sob as mais diversas formas, tanto durante o seu processo de desenvolvimento como durante sua operação e, portanto, vida útil.

Confiabilidade Conceitual refere-se às características que tornam o produto confiável para seus usuários, do ponto de vista de seu conteúdo, satisfazendo os requisitos que motivaram a sua construção.

Confiabilidade da representação refere-se às características que tornam o produto confiável para seus usuários considerando-se aspectos relativos à sua forma, e que tornam possível sua compreensão e manipulação, tendo em conta as diversas representações do produto desde a especificação de requisitos até o próprio código. Esta característica torna possível que o software seja entendido e manipulado por seus diferentes tipos de usuários durante o período de desenvolvimento e de vida operacional.

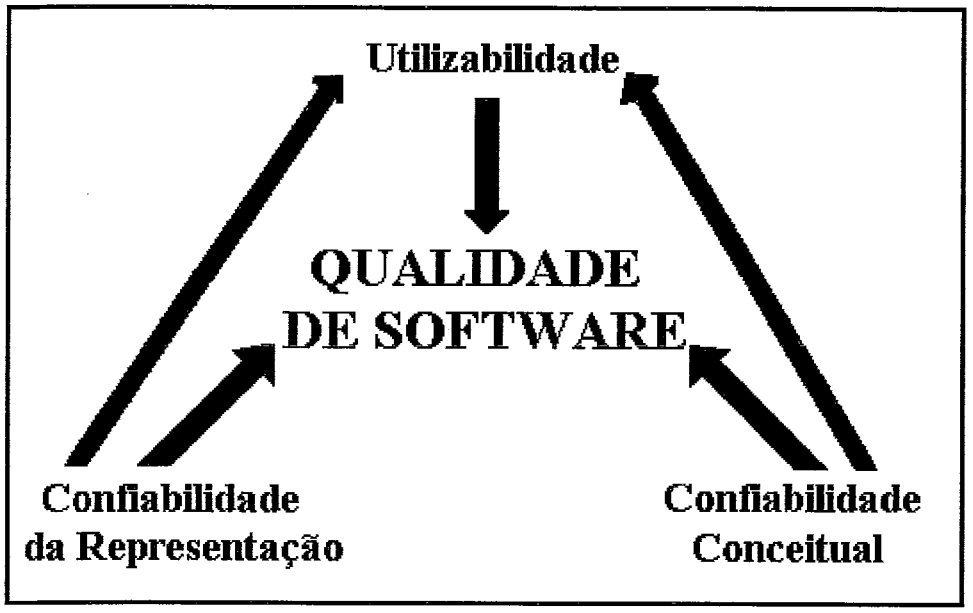

Figura 4.1 - Objetivos para atingir qualidade de software

De acordo com o modelo proposto por Rocha [Rocha 83], que descrevemos no capítulo 2, os objetivos de qualidade são atingidos através de fatores que por sua vez podem ser compostos por subfatores. Fatores e sub-fatores são avaliados através de critérios, para os quais devem ser definidos processos de avaliação.

Nas seções que se seguem, definimos os fatores e subfatores através dos quais são atingidos cada um dos objetivos de qualidade considerando sistemas especialistas de uma forma geral.

Os critérios e processos de avaliação foram definidos para o caso específico de sistema especialista para diagnóstico em Cardiologia, que descrevemos no próximo

caítulo, considerando os diferentes produtos gerados ao longo do processo de desenvolvimento. Esta opção, faz-se necessária porque os critérios e, consequentemente, os processos de avaliação a eles relacionados são extremamente dependentes do domínio da aplicação e, até mesmo, do projeto em questão.

4.2.1 Objetivo: Utilizabilidade

Utilizabilidade é um objetivo fundamental que se refere às diversas formas como um software pode ser utilizado durante seu desenvolvimento e vida útil.

Um produto de software é, normalmente, utilizado durante o seu desenvolvimento com as seguintes finalidades:

- manutenção dos produtos intermediários (especificações e projeto) para incorporação de novos requisitos, funções, objetos e/ou conhecimentos;
- avaliação dos produtos gerados ao longo do desenvolvimento;
- realização de modificações, resultantes das avaliações realizadas;
- implementação do produto a partir de suas especificações e projeto.

Durante sua vida operacional o produto deve poder ser utilizado por seus diferentes usuários facilitando suas tarefas, e atendendo aos requisitos que motivaram a sua construção. Para isto, o produto deve operar de forma eficiente e lucrativa, e deve ser compatível com outros sistemas em operação no mesmo ambiente e com os quais precisa interagir. Além disso, deve ser possível convertê-lo para outro ambiente de hardware/software quando isto se fizer necessário.

Estas formas de utilização do software, sugerem os atributos de qualidade que ele deve possuir de forma a satisfazer as necessidades de seus usuários. Assim sendo consideramos que este objetivo é atingido através dos fatores *manutenibilidade*, *reutilizabilidade*, *facilidade de utilização*, *compatibilidade*, *portatilidade*, *eficiência*, *rentabilidade*, *adequação tecnológica*, *aplicabilidade* e *implementabilidade* (Figura 4.2).

Considerando-se estes fatores, e os critérios a eles relacionados, é possível avaliar se estas características estão sendo alcançadas ao longo do processo de desenvolvimento e estão presentes no produto final.

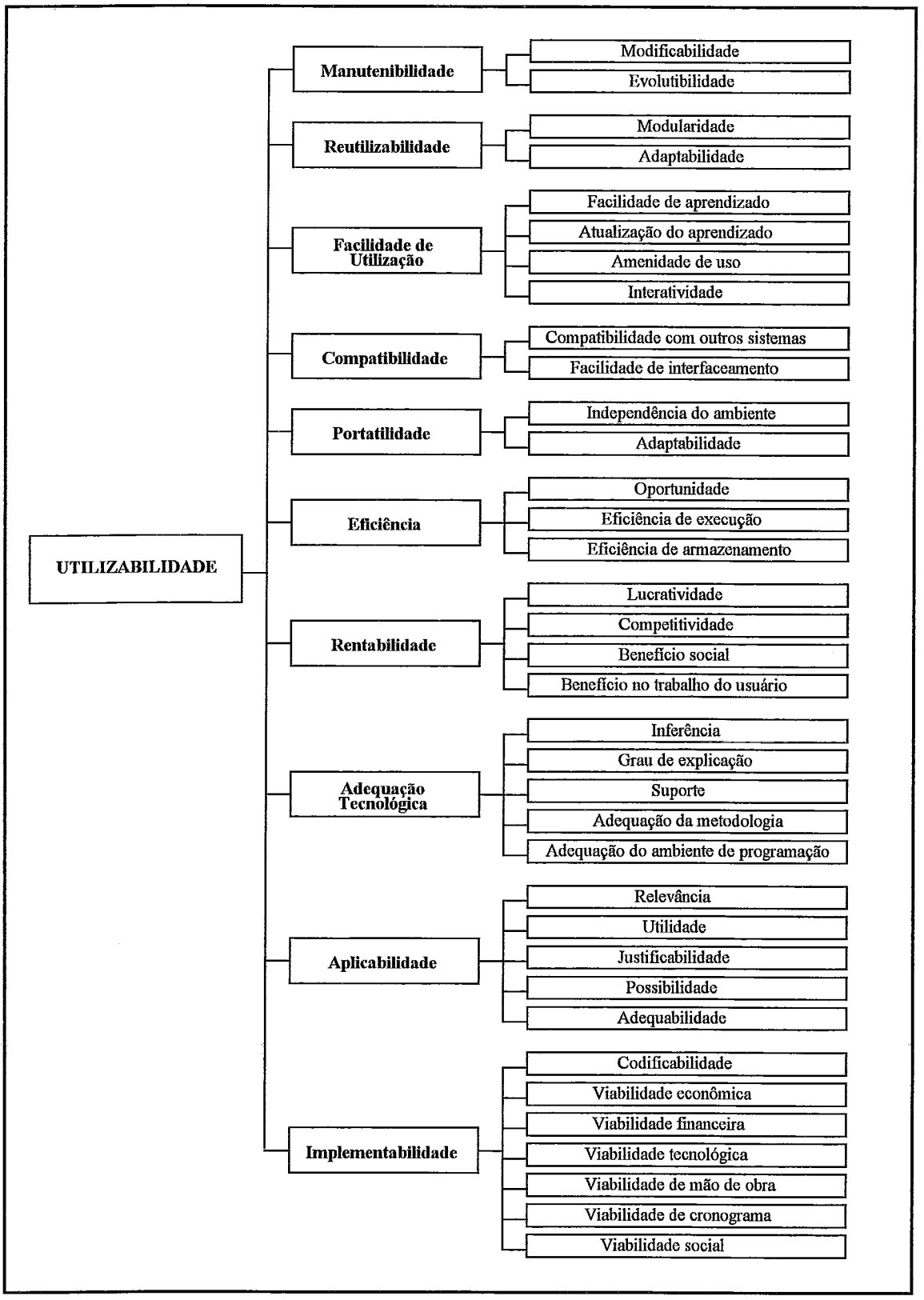

Figura 4.2 - Fatores e sub-fatores relacionados à Utilizabilidade

Muito destas características são pertinentes a qualquer tipo de sistema. Sendo, portanto, também, relevantes no caso de sistemas especialistas. Mas algunas delas são específicas de sistemas especialistas. É importante, por exemplo, para qualquer tipo de sistema, garantir que possa ser facilmente modificável e evolutível, mas no caso de sistemas especialistas a evolutibilidade é uma característica fundamental pois a base de conhecimento sofre normalmente um refinamento progressivo para se tornar cada vez mais completa.

A reutilização é também importante em qualquer tipo de software e no caso específico de sistemas especialistas a reutilização de qualquer produto para outro sistema especialista que possa ter até uma atividade diferente é de grande utilidade. Por exemplo, a base de conhecimento referente a uma atividade de interpretação ou diagnóstico de um assunto de algum domínio de problema pode ser reutilizada para tutorias sobre como diagnósticar ou interpretar o mesmo assunto do domínio considerado.

A facilidade de utilização reflete a preocupação com o fato de que os usuários destes sistemas algumas vezes não estão familiarizados com o domínio do problema e seu jargão, podendo, ainda, não estar familiarizados com o próprio uso de computadores na realização de suas tarefas. Estes usuários necessitam, portanto, que o sistema seja de uso amigável. Além disso, os sistemas especialistas tem a característica de ter que manter um diálogo com o usuário para que possa inferir suas conclusões. Esta interação deve portanto ser o mais clara, objetiva e concisa possível.

O atributo compatibilidade, que avalia o interfaceamento do sistema com outros produtos de software, e a portatilidade, que torna o produto independente de ambientes específicos de hardware e software, é relevante para sistemas especialistas, assim como para qualquer outro sistema.

O atributo eficiência refere-se à preocupação com o melhor uso dos recursos disponíveis de forma a prover resultados em tempo hábil dado que para muitos dos sistemas especialistas tem-se um tempo crítico para a apresentação da solução.

A avaliação da rentabilidade do produto pode justificar ou não o seu desenvolvimento. No caso de sistemas especialistas, o benefício no trabalho do usuário é bastante típico visto que este sistemas se caracterizam em grande parte por atividades de apoio ao usuário.

A avaliação segundo o atributo adequação tecnológica garante que os aspectos peculiares de sistemas especialistas (como tratamento de incerteza e mecanismo de inferência) estão sendo adequadamente utilizados. A aplicabilidade avalia se a tarefa é

típica de ser resolvida por um sistema especialista, se é relevante o desenvolvimento do sistema para a tarefa que vai desempenhar e se será realmente útil. Além disso é necessário avaliar se é justificável, possível e adequado o seu desenvolvimento. Segundo Waterman [Waterman 86] o desenvolvimento de sistemas especialistas só é justificável se a tarefa traz alto reforno finaceiro, se a especialidade está escassa, ou quando a especialidade é necessária em diferentes locais ou em ambientes hostis.

No caso da implementação do sistema é necessário averiguar se está se utilizando técnicas de codificação adequadas para o sistema especialista que está sendo desenvolvido.

Assim sendo, tem-se os seguintes fatores e subfatores de qualidade relacionados ao objetivo utilizabilidade, quando consideramos produtos de software do tipo sistemas especialistas:

Fator: Manutenibilidade - refere-se à facilidade com que, sempre que necessário, podem-se realizar alterações no sistema. É atingido pelos sub-fatores: *modificabilidade* e *evolutibilidade*.

Subfator: Modificabilidade - característica que um sistema deve ter de forma a facilitar a realização de modificações.

Subfator: Evolutibilidade - característica que o sistema deve ter de forma a permitir sua própria evolução, através de refinamentos sucessivos que representem o conhecimento de forma cada vez mais completa.

Fator: Reutilizabilidade - refere-se a característica do sistema, ou parte dele, em qualquer de suas formas, possa ser reutilizado por outros sistemas. É atingido pelos subfatores: *modularidade* e *adaptabilidade*.

Subfator: Modularidade - característica do sistema ser projetado e implementado através de uma estrutura de módulos independentes e particionados logicamente.

Subfator: Adaptabilidade - característica do sistema ser fácil de adaptar para permitir sua reutilização por outro sistema.

Fator: Facilidade de utilização - refere-se às características do sistema que o tornam de fácil uso e aprendizado por seus usuários. É atingido pelos sub-fatores: *facilidade de aprendizado, atualização do aprendizado, amenidade de uso* e *interatividade*.

Subfator: Facilidade de aprendizado - refere-se à facilidade para entendimento e utilização do sistema por usuários finais iniciantes.

Subfator: Atualização do aprendizado - refere-se à facilidade para manter o aprendizado atualizado após terem sido realizadas modificações no sistema.

Subfator: Amenidade de Uso - característica do sistema apresentar os resultados de forma clara, ser capaz de disponibilizar auxílio de forma rápida e apresentar estabilidade no uso.

Subfator: Interatividade - característica do sistema manter um diálogo conciso e claro com o usuário final.

Fator: Compatibilidade - refere-se às características que fazem o sistema ser compatível com outros sistemas da organização. É atingido pelos sub-fatores: *compatibilidade com outros sistemas* e *facilidade de interfaceamento*.

Subfator: Compatibilidade com outros sistemas - característica do sistema interagir com outros sistemas da organização, complementando informações e tecnologias.

Subfator: Facilidade de interfaceamento - característica do sistema oferecer facilidades na interação com outros sistemas obedecendo a padronizações estabelecidas.

Fator: Portatilidade - refere-se às características que permitem que o sistema seja operado em configurações de equipamentos diferentes da original. É atingido pelos sub-fatores: *independência do ambiente* e *adaptabilidade*.

Subfator: Independência do ambiente - característica do sistema não possuir restrições para execução em configurações de hardware diferentes das originais.

Subfator: Adaptabilidade - característica do sistema ser fácil de adaptar para permitir sua extensão em configurações de hardware diferentes das originais.

Fator: Eficiência - refere-se à característica do sistema ser executado com a melhor utilização possível de recursos. É atingido pelos sub-fatores: *oportunidade*, *eficiência de execução* e *eficiência de armazenamento*.

Subfator: Oportunidade - característica do sistema produzir resultados, em situações típicas, no tempo adequado estabelecido na especificação de requisitos.

Subfator: Eficiência de execução - característica do sistema ser executado no menor tempo possível.

Subfator: Eficiência de armazenamento - característica do sistema usar adequadamente a memória disponível.

Fator: Rentabilidade - refere-se aos benefícios financeiros e sociais obtidos com a utilização do sistema. É atingido pelos sub-fatores: *lucratividade*, *competitividade*, *benefício social* e *benefício no trabalho do usuário*.

Subfator: Lucratividade - característica do sistema oferecer retorno financeiro com sua utilização e/ou comercialização.

Subfator: Competitividade - característica do sistema aumentar o nível de competitividade da organização.

Subfator: Benefício social - característica do sistema oferecer retorno social com sua utilização e/ou comercialização.

Subfator: Benefício no trabalho do usuário - característica da utilização do sistema oferecer uma simplificação e melhoria nas condições de trabalho de seus usuários.

Fator: Adequação tecnológica - está relacionado às características e à adequação da tecnologia utilizada para construção do sistema. É atingido pelos sub-fatores: *inferência, grau de explicação, suporte, adequação da metodologia* e *adequação do ambiente de programação.*

Subfator: Inferência - característica do sistema ter um mecanismo de raciocínio sofisticado.

Subfator: Grau de explicação - característica do sistema prover uma justificativa da solução gerada.

Subfator: Suporte - característica do sistema ser provido de técnicas auxiliares de suporte como, tratamento de incerteza, verificação da consistência e de concorrência e compilação de regras.

Subfator: Adequação da metodologia - característica do sistema usar procedimentos e métodos adequados para a representação do conhecimento.

Subfator: Adequação do ambiente de programação - característica do sistema ter sido implementado utilizando um ambiente de programação adequado às suas necessidades e que seja capaz de suportar futuras evoluções no sistema.

Fator: Aplicabilidade - refere-se às características do sistema que o fazem adequado e útil no contexto em que está inserido. É atingido pelos sub-fatores: *relevância, utilidade, justificabilidade, possibilidade* e *adequabilidade.*

Subfator: Relevância - característica da utilização do sistema ser importante para a realização da tarefa envolvida.

Subfator: Utilidade - característica da utilização do sistema ter consequências significativamente úteis.

Subfator: Justificabilidade - característica do sistema ter sua construção justificada pelo contexto em que estará inserido.

Subfator: Possibilidade - característica de ser possível a construção do sistema, considerando-se a existência de especialistas e as tarefas envolvidas.

Subfator: Adequabilidade - característica de ser adequada a construção do sistema, considerando-se a natureza, a complexidade e o escopo do problema.

Fator: Implementabilidade - refere-se às características que tornam viável a implementação do sistema. É atingido pelos sub-fatores: *codificabilidade, viabilidade econômica, viabilidade financeira, viabilidade tecnológica, viabilidade de mão-de-obra, viabilidade de cronograma* e *viabilidade social.*

Subfator: Codificabilidade - característica do sistema utilizar técnicas de representação do conhecimento, para codificação e atualização da base de conhecimentos, poderosas e flexíveis.

Subfator: Viabilidade Econômica - viabilidade do sistema poder ser construído com uma relação custo/benefício aceita pelos usuários e desenvolvedores.

Subfator: Viabilidade Financeira - viabilidade do sistema poder ser construído mediante a disponibilização do capital necessário para o seu desenvolvimento.

Subfator: Viabilidade tecnológica - viabilidade do sistema poder ser construído considerando-se a existência e a disponibilização da tecnologia necessária para o seu desenvolvimento.

Subfator: Viabilidade de mão de obra - viabilidade do sistema poder ser construído, considerando-se a existência e a disponibilização da mão de obra necessária para o seu desenvolvimento.

Subfator: Viabilidade de cronograma - viabilidade do sistema poder ser construído dentro do limite de tempo planejado, considerando possíveis ocorrências de imprevistos e com flexibilidade para introdução de atividades não projetadas e/ou contingenciais, mantendo a qualidade definida para o produto.

Subfator: Viabilidade social - viabilidade do sistema poder ser construído considerando-se o grau de satisfação de seus futuros usuários, bem como os impactos gerados sobre o sistema social ao qual servirá.

4.2.2 Objetivo: Confiabilidade da Representação

Durante o seu desenvolvimento e vida operacional, produtos de software são manipulados por diferentes usuários, em geral, diferentes de seus desenvolvedores e que muitas vezes não tem qualquer possibilidade de contato com os mesmos. O objetivo confiabilidade da representação refere-se aos aspectos necessários para facilitar a compreensão e manipulação do software sob suas diversas formas de representação (especificação, projeto e código). Dessa forma, os atributos identificados são necessário em qualquer tipo de sistema, inclusive, sistemas especialistas.

Ao se considerar o caso específico de sistemas especialistas é importante se ter em conta que, à medida em que o conhecimento vai sendo elicitado, ele deve ser representado de forma a poder ser compreendido e validado pelos especialistas envolvidos no seu desenvolvimento. Para isso, além de prover uma documentação compreeensível é importatente procurar organizar o conhecimento de uma forma hierárquica que possa facilitar a manipulação e avaliação do conhecimento pelos especialistas. Estes aspectos, também, são essenciais nas futuras evoluções do sistema, uma característica típica de sistemas especialistas.

Para que seja possível um real envolvimento dos especialistas no desenvolvimento do sistema, é necessário que a documentação gerada ao longo do desenvolvimento seja facilmente compreensível pelos mesmos e fácil de ser manipulada, de forma a se poder encontrar, em cada momento, a informação desejada. Assim sendo, este objetivo é atingido através dos fatores *legibilidade* e *manipulabilidade* (Figura 4.3).

Fator: Legibilidade - refere-se às características do sistema que o tornam de fácil compreensão. É atingido pelos sub-fatores: *clareza*, *concisão*, *estilo*, *modularidade*, *correção da representação*, *uniformidade da terminologia*, *uniformidade no grau de abstração* e *completude da documentação*.

Subfator: Clareza - característica do sistema estar especificado, modelado e implementado da forma mais clara possível, isento de práticas que o tornem complexo e de difícil entendimento.

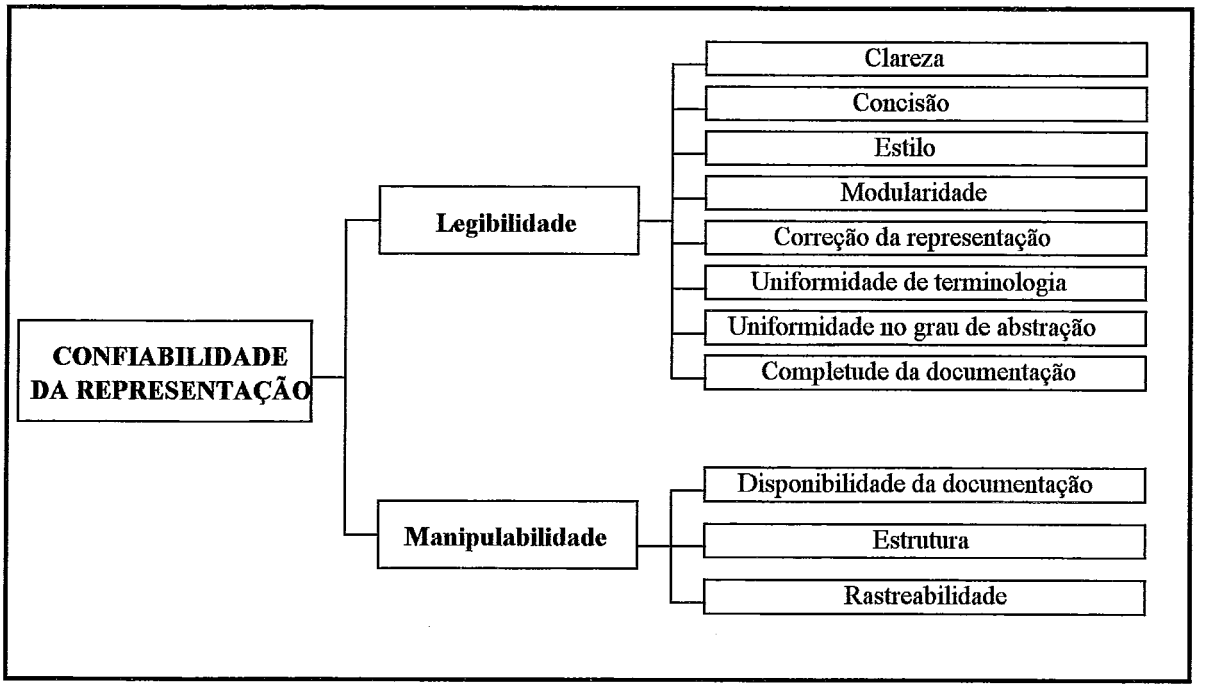

Figura 4.3 - Fatores e sub-fatores relacionados à Confiabilidade da Representação

Subfator: Concisão - característica do sistema ter sido especificado e modelado de forma concisa e implementado com a quantidade mínima de comandos.

Subfator: Estilo - refere-se à necessidade da especificação, modelagem e programas do sistema conterem elementos adequados de estilo, de forma que expressem seus objetivos e especificações de maneira simples, elegante, organizada e direta, considerando as padronizações e/ou recomendações estabelecidas no processo de desenvolvimento adotado.

Subfator: Modularidade - característica do sistema ser projetado e implementado através de uma estrutura de módulos independentes e particionados logicamente.

Subfator: Correção da representação - característica do sistema estar correto do ponto de vista do uso da linguagem de especificação adotada, no que se refere a notação, semântica, sintaxe e formato de documentação.

Subfator: Uniformidade de terminologia - característica do sistema estar documentado com uniformidade de notação e padronização de termos técnicos.

Subfator: Uniformidade no grau de abstração - característica da documentação do sistema possuir um nível uniforme de detalhe, considerando-se um determinado estágio do desenvolvimento.

Subfator: Completude da documentação - característica da documentação estar completa de acordo com os roteiros estabelecidos no Plano de Documentação do projeto.

Fator: Manipulabilidade - refere-se às características que tornam possível, ao se analisar o sistema, encontrar, com facilidade, as informações desejadas. É atingido pelos sub-fatores: *disponibilidade da documentação, estrutura* e *rastreabilidade*.

Subfator: Disponibilidade da documentação - característica do sistema ter sua documentação atualizada e pronta para uso quando necessário.

Subfator: Estrutura - característica do sistema possuir um padrão definido de composição de suas partes, formando uma organização hierárquica de regras e/ou conhecimento.

Subfator: Rastreabilidade - característica do sistema possuir uma documentação e código que permitam a busca de informações através da sequência de agregação de detalhes de um determinado aspecto, desde sua visão mais geral até a mais detalhada, e vice-versa.

4.2.3 Objetivo: Confiabilidade Conceitual

Confiabilidade conceitual é um objetivo de grande importância para a qualidade de um produto pois refere-se às características que garantem que o software corresponde ao que se propõe. Avaliar a confiablidade conceitual significa, portanto, avaliar o produto com relação ao seu conteúdo.

A avaliação da confiabilidade conceitual no caso de sistemas especialistas significa buscar garantir que tanto a documentação gerada ao longo do desenvolvimento quanto a base de conhecimento correspondem ao conhecimento elicitado do especialista. Nesse sentido, portanto, destacamos a importância de se ter uma base de conhecimento o mais completa possível, que gere soluções para todos os casos típicos, não tenha conhecimento redundante ou contraditório e que corresponda, o mais que possível, ao conhecimento do especialista humano, já que o sistema vai atuar como tal. Este objetivo é, portanto, atingido através dos fatores *fidedignidade* e *integridade* (Figura 4.4).

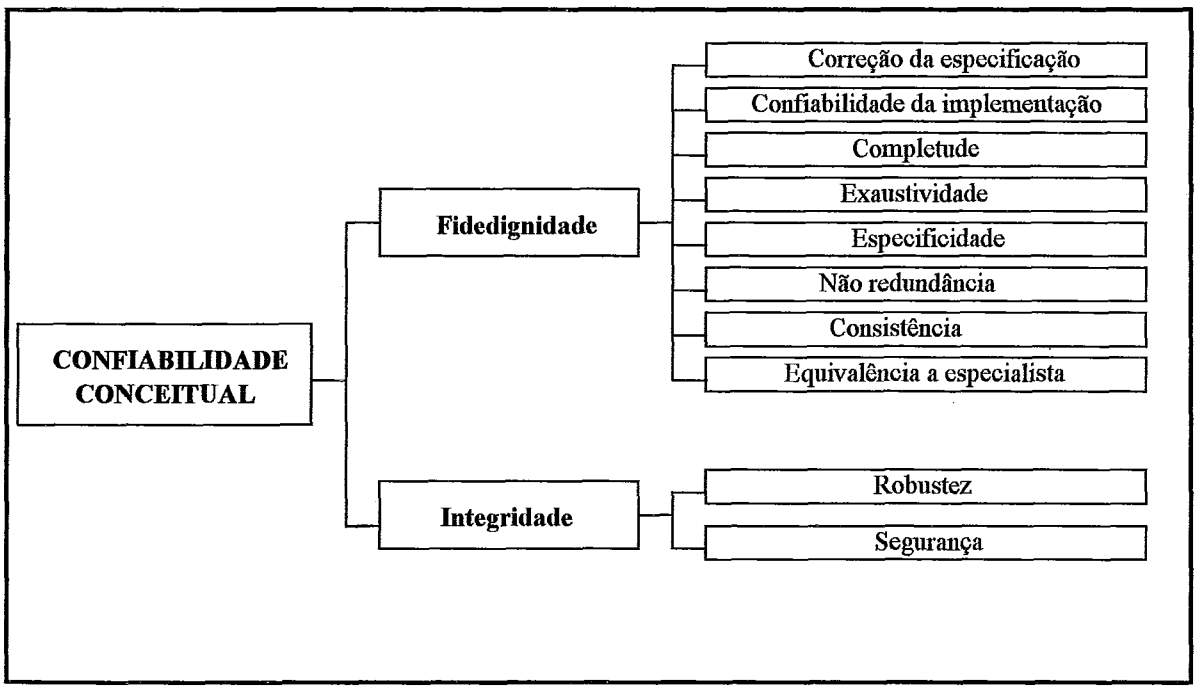

Figura 4.4 - Fatores e sub-fatores relacionados à Confiabilidade Conceitual

Fator: Fidedignidade - refere-se às características que fazem o sistema corresponder aos requisitos especificados pelo usuário. É atingido pelos sub-fatores: *correção da especificação, confiabilidade da implementação, completude, exaustividade, especificidade, não redundância, consistência* e *equivalência ao especialista.*

Subfator: Correção da especificação - característica da especificação do sistema representar corretamente o conhecimento elicitado do especialista.

Subfator: Confiabilidade da implementação - característica do sistema proporcionar soluções únicas e suficientemente acuradas de modo a satisfazer a utilização pretendida pelo usuário.

Subfator: Completude - característica do sistema possuir somente informações utilizáveis, garantindo a ausência de atributos não referenciados, conclusões não alcançáveis e partes do domínio não cobertas.

Subfator: Exaustividade - característica do sistema ter uma base de conhecimento com um nível de riqueza que permita, sempre, a geração de resultados confiáveis.

Subfator: Especificidade - característica da especificação do sistema abordar, apenas os requisitos necessários e de sua implementação gerar, apenas, as soluções especificadas.

Subfator: Não redundância - característica da especificação do sistema não conter conhecimento nem gerar soluções redundantes.

Subfator: Consistência - característica do sistema não apresentar conhecimentos contraditórios, nem soluções contraditórias em sua implementação.

Subfator: Equivalência a especialista - característica do sistema ter um desempenho com confiabilidade equivalente a de um especialista, tanto em termos de escopo do problema como da solução gerada.

Fator: Integridade - refere-se às características que tornam o sistema capaz de enfrentar situações hostis. É atingido pelos sub-fatores: *robustez* e *segurança*.

Subfator: Robustez - característica do sistema ser capaz de enfrentar situações hostis, reagindo a elas sem perda de controle.

Subfator: Segurança - características do sistema ser confiável sob o ponto de vista de seu uso, sem risco de provocar danos materiais e/ou de vidas humanas, protegendo-se de violações e garantindo a privacidade.

4.3 Conclusão

Neste capítulo identificamos características de qualidade de software, quando este é um sistema especialista. No próximo capítulo descrevemos uma experiência na avaliação da qualidade de um sistema especialista para diagnóstico médico na área de cardiologia onde se partiu das características de qualidade definidas neste capítulo.

Capítulo 5

Avaliação da Qualidade de um Sistema Especialista em Cardiologia

"A tecnologia não salva vidas, mas pessoas com a ajuda da tecnologia podem salvar vidas"

Belden et al [1]

5.1 Introdução

Sistemas especialistas vem sendo aplicados em diferentes domínios de aplicação nas atividades de interpretação, predição, projeto, planejamento, monitoração, prescrição, reparo, instrução, controle e diagnóstico [Waterman 86], [Luger 89], [Aguiar 92], [Assis 92]. As características de qualidade, apresentadas no capítulo anterior, foram definidas sem considerar a área da aplicação e a atividade específica a ser desempenhada pelo sistema especialista.

Neste capítulo apresentamos uma experiência de avaliação de software, segundo essas características de qualidade, a partir do desenvolvimento de um sistema especialista para diagnóstico em Cardiologia, o projeto SEC, que está sendo realizado na Fundação Bahiana de Cardiologia (FBC). Consideramos, portanto neste caso, um domínio de aplicação específico e um projeto real onde o sistema tem o objetivo de realizar a atividade de apoio ao diagóstico.

A realização deste trabalho teve três objetivos:

- validar a adequação das características definidas no capítulo anterior;
- particularizar estas características para o projeto SEC, e,
- validar idéias desta tese quanto à avaliação da qualidade de sistemas especialistas participando do desenvolvimento e avaliação de um projeto concreto.

[1] T.Belden et al; "Developing an Advanced Tool for the Clinician: using industrial design and interface design together to bring technology into the hands of the user"; Computer-Based Medical Systems, IEEE Computer Society Press; 1993

Para situar o problema do desenvolvimento e avaliação da qualidade de um sistema especialista para diagnóstico em Cardiologia, iniciamos o capítulo abordando questões atuais relacionadas ao desenvolvimento de software médico, em geral e no caso específico de sistemas especialistas.

5.2 A Questão do Software Médico

Com o rápido crescimento da tecnologia de computadores, muitas organizações tem colocado grandes exigências nos produtos de software. Estas exigências muitas vezes estão relacionadas a produtos de software com missão de exercer controle total ou parcial sobre funções críticas, como controle ou monitoração de dispositivos médicos, controle de dispositivos de tráfico e sistemas de defesa.

Nesse contexto, a infraestrutura de informática em sistemas de saúde tem se apresentado com uma significativa evolução nos últimos anos. Segundo Kim [Kim 93] estes softwares podem ser divididos em três tipos: software de controle de dispositivos médicos, software de processo e software médico. Os primeiros, muitas vezes complexos, referem-se aos softwares embutidos no hardware que podem controlar, regular ou operar com dispositivos médicos (como, por exemplo, sistemas que planejam terapia de tratamento de radiação). Os softwares de processo podem ser utilizados na produção, montagem ou teste de produtos médicos como dispositivos e produtos farmacêuticos. E, finalmente, softwares médicos são aqueles que desempenham funções médicas independentes do hardware utilizado.

Neste sentido muito se tem produzido em vários países, incluindo o Brasil: sistemas de controle da banco de sangue [Kim 93], [Fernandes 94], sistemas de reconhecimento de voz [Prosdocimo 94], [Pecoits 94], sistemas de imagem para quantificação não invasiva do volume de sangue ventricular [James 93], sistema para realce e segmentação de imagens médicas [Silva 94a], hiperdocumentos para documentos de cateterismo cardíaco [Ip 93], sistema multimidia de conferência para diagnótico médico cooperativo [Ng 93], sistema para gerenciamento de informação durante cirurgia cardíaca através da coleta de dados em tempo real [Tisdell 93], análise de imagens tridimensionais de hemorragia intracerebral [Dhawan 93] e lesões do cérebro [Kozinsko 94], sistemas para auxílio ao planejamento de neurocirúrgias através da análise de imagens do cérebro [Furukawn 94], sistemas para avaliação de funções cardíacas de crianças pela análise de imagens de ecocardiografia [Robert 94], sistemas hipermídia inteligente [Jao 93], [Ma 93], tutores [Klemt 94], [Morna 94], [Fontoura 94], sistema com utilização de redes neurais para interpretação de eletrocardiograma [Spitzer 93], sistemas inteligentes de

registros médico [Chen 93], sistemas para classificação internacional de doenças [Kaempf 94], sistemas de informações médicas [Chao 93], [Coutinho 94], [Pryor 94], sistema de prescrição [Silva 94b], sistema de controle de infecções hospitalares [Hoefel 94], [Deon 94], sistemas de apoio à decisão [Menezes 94] além de vários sistemas especialistas que serão relacionados na próxima seção.

Sabemos que todo software está sujeito a falhas. Infelizmente, certas falhas nos softwares de aplicações médicas podem resultar em consequências catastróficas como lesões, graves danos pessoais ou, até, mesmo perdas de vidas humanas. Tem-se o exemplo do Therac 25, máquina com controle computadorizado de terapia de radiação, que foi responsável pela morte de dois pacientes e causou lesões em outros trinta, devido a um erro no software que levou estes pacientes a receberem aproximadamente 125 vezes a dose terapêutica normal de radiação. Devido a problemas como este, o governo, a indústria e pesquisadores acadêmicos estão procurando novos caminhos para prevenir falhas no desenvolvimento e certificação de softwares críticos [Gowen 93], [Levenson 90].

Preocupado com a utilização da tecnologia de computadores na área médica, o *Food and Drug Administration* (FDA), órgão responsável pela fiscalização de produtos médicos nos Estados Unidos, estabeleceu um regulamento para softwares como produtos com aplicações médicas ou componentes de dispositos médicos regulamentados. O FDA tem autoridade de regulamentar os três tipos de software: os software de dispositivos médicos tratando-os como componentes, parte ou acessórios de dispositivos médicos, softwares de processo que estão sujeitos a regulamentação dos requisitos da *Good Manufacturing Practice* (GMP) da FDA para dispositivos médicos que associado às industrias procuram educar os produtores de dispositivos nos requisitos regulamentados; e os softwares médicos através de uma fiscalização de produtos de software. [Kim 93]

Como a grande maioria das falhas de dispositivos relacionadas a software são atribuídas a erros relacionados ao projeto, é dada uma grande importância à necessidade de se terem projetos bem elaborados e com procedimentos para garantia de qualidade. Para lidar com as dificuldades de avaliação da qualidade de software, grupos da FDA tem dado uma forte ênfase à utilização de um processo de desenvolvimento de software racional e bem documentado, além de práticas eficazes para garantia da qualidade [Kim 93].

Dessa forma muitos estudos e experiências tem sido realizados no desenvolvimento de softwares para aplicações médicas. Para que este desenvolvimento gere produtos de sucesso é necessário um completo entendimento do objetivo e das necessidades do

usuário [Eureka 93], [Belden 93]. Nesse sentido, foi experimentado o processo de *Quality Function Deployment* (QFD) no desenvolvimento de softwares de processo. QFD aplica princípios da análise de função para detalhar atividades de garantia da qualidade, diferindo das outras abordagens pelo enfoque no entendimento dos requisitos do usuário. Dessa forma, características e funções críticas para satisfação do usuário são projetadas, falhas potenciais podem ser antecipadas e analisadas e há uma economia de tempo e dinheiro [Mazur 93].

É certo que assim como nos sistemas tradicionais, a produção de software para aplicações médicas de qualidade é obtida através do uso de técnicas de Engenharia de Software adequadas e de procedimentos de verificação e validação (V&V) durante todas as fases de desenvolvimento [Levenson 90], [Mallory 93], [Morozoff 94]. O nível das atividades de V&V aplicadas a estes softwares é determinada pelo uso pretendido para o software e o risco de causar sérias lesões ou morte [Leffingwell 93]. De acordo com este nível de avaliação os softwares para dispositivos médicos são classificados como nível baixo, moderado ou alto. Por exemplo, os softwares que fornecem aos médicos informações sobre tratamento são, em geral, classificados como moderados. Entretanto, embora um número crescente de sistemas especialistas para uso em especialidades clínicas estejam sendo desenvolvidos, não existem indicações de que eles representem risco para vidas humanas, pois estes sistemas caracterizam-se por atuarem como um consultor, havendo sempre intervenção humana antes que possa ocorrer qualquer impacto sobre o paciente [Kim 93].

Boegh [Boegh 93] propõe uma classificação geral para o nível em que deve ser feita a avaliação de produtos de software relacionada ao grau de risco que a aplicação oferece. Estabelece quatro níveis que implicam num conjunto crescente de procedimentos de avaliação em termos de profundidade e detalhamento e que, portanto, fornecem graus diferentes de confiança na qualidade do produto de software. A Figura 5.1 mostra algumas indicações de em que nível, produtos de software devem ser avaliados considerando-se o tipo de aplicação, onde D é o nível mais baixo e A o mais alto. De acordo com esta classificação de Boegh, produtos de software para aplicações médicas devem ter um alto nível de avaliação, sendo superado, apenas, para os softwares que lidam com problemas nucleares.

A questão da garantia da qualidade de softwares para dispositivos médicos é, fortemente, direcionada para preocupações com a segurança e a confiabilidade destes softwares [Elahi 93], [Gowen 93], [Voas 93]. Para Elahi [Elahi 93] a preocupação com a segurança oferecida pelo produto deve acontecer desde a fase de definição de requisitos e durante todo o ciclo de desenvolvimento do produto. Dessa forma, ele propõe técnicas

de análise de segurança que avaliem os efeitos de todos os potenciais erros que poderiam contribuir para um desempenho indesejável de uma função crítica.

NÍVEL	APLICAÇÕES	RISCO PARA O AMBIENTE	RISCO PARA PESSOAS	RISCO FINANCEIRO
D	Sistemas de diversão	Pequenos danos para os proprietários	Nenhum risco para pessoas	Perda econômica insignificante
C	Alarme de incêndio Controle de processo	Danos para os proprietários	Algumas pessoas prejudicadas	Perda econômica significativa
B	Sistemas médicos Sistemas Financeiros	Danos ambientais recuperáveis	Risco para vidas humanas	Grande perda econômica
A	Sistemas de estradas de ferro Sistemas nucleares	Danos ambientais irrecuperáveis	Muitas mortes	Desastre finaceiro

Figura 5.1 - Guia para seleção do nível de avaliação [adaptada de Boegh 93]

Collins [Collins 94], no artigo "How Good is Good Enough?" usa como exemplo para justificar sua proposta de que a ética no desenvolvimento de software está na questão de sua qualidade, com uma situação referente à segurança de um software de prescrição de medicamentos de um hospital. Segundo este autor, existem responsabilidades éticas para todas as pessoas envolvidas tanto no processo de desenvolvimento quanto de utilização do software: os desenvolvedores do software, os compradores do software (o hospital) e os usuários (médicos, enfermeiros, etc). O autor refere-se ainda, às pessoas que nem chegam a ter contato com o software, mas que são afetadas pelo mesmo. Collins chama este grupo de pessoas, de "penumbra". No caso de software de aplicações médicas, neste grupo estão os pacientes. O autor define, ainda, as obrigações de cada grupo para com o outro. Quando considera o grupo "penumbra", maior preocupação nos softwares de aplicações médicas, Collins afirma que os desenvolvedores tem obrigação de oferecer proteção contra danos físicos e financeiros que possam ser causados pela aplicação, que os compradores de produtos de software devem comprar e colocar em uso apenas aqueles softwares que garantam a máxima segurança, que os usuários devem ter o máximo cuidado em reduzir o perigo de

qualquer risco para o público e que o próprio grupo "penumbra" deve ser consciente das limitações do software e exigir leis com relação à qualidade de software.

Hoje se considera fundamental o papel do software na área médica, tanto controlando dispositivos médicos, quanto controlando processos ou apoiando médicos e enfermeiros em seu trabalho. Com isto espera-se melhorar todo o sistema de saúde no que diz respeito à qualidade, custos, capacidade funcional, bem estar e satisfação do paciente com o atendimento [Banta 92], [Elahi 93],[Hayes-Roth 94].

Neste contexto é amplamente reconhecido o papel da Engenharia de Software, tanto no que se refere ao processo de desenvolvimento de software quanto à garantia da qualidade dos produtos. O objetivo da Engenharia de Software, considerando-se aplicações médicas, é, portanto, estabelecer processos de desenvolvimento e controle da qualidade, capazes de produzir software de alta qualidade e confiabilidade, que não coloquem em risco vidas humanas mas ao contrário contribuam para a melhoria do atendimento médico-hospitalar [Kim 93], [Voas 93], [Zemlin 93], [Linberg 93], [Collins 94].

5.3 Sistemas Especialistas Médicos

O crescente uso de sistemas computadorizados em medicina tem destacado a importância da inteligência artificial e, em particular, de sistemas especialistas para esta área [Kim 93], [Hayes-Roth 94]. Segundo Kane [Kane 88] quando questionamos quais as tarefas que um sistema médico computadorizado poderia fazer, verificamos que tratar de pacientes corresponde à formulação de diagnóstico e administração adequada de tratamento. Pesquisas e experiências de desenvolvimento de sistemas especialistas tem sido progressivamente realizadas para alcançar estes objetivos.

Saranimmi [Saranimmi 91] observa que apesar de muitas pesquisas e esforço de desenvolvimento estarem sendo investidos em sistemas especialistas médicos, ainda há muito a ser pesquisado no que se refere ao entendimento de como projetar, construir, verificar, validar e atualizar estes sistemas. Observa, ainda, a necessidade de um melhor entendimento da área médica por parte dos engenheiros de software e, especialmente, de entendimento das funções das pessoas e computadores neste ambiente. Saramini relata uma pesquisa realizada por Potthoff[2] em 64 sistemas onde a maioria refere-se a diagnóstico de tarefas altamente estruturadas e que poucos estão projetados para suporte

[2] P. Potthoff et al.; "Expert Systems in Medicine"; Int. J. Technol. Assess, Health Care 4 ;1988

à terapia em domínios muito específicos e, ainda, que o treinamento das pessoas para a utilização destes sistemas geralmente não existe.

Segundo Saranimmi dois problemas devem ser bem discutidos para se ter um futuro promissor na utilização de sistemas especialistas na área médica. O primeiro problema refere-se ao entendimento de quais locais, realmente, servem aos propósitos do suporte à decisão médica por computadores e especialmente por sistemas especialistas. Para isso, dois aspectos devem ser considerados. O primeiro, é o papel das pessoas e computadores na solução de problemas onde ambos são perfeitamente utilizados: o computador deve apoiar as pessoas, como um complemento e não como substituto. O segundo, refere-se à estrutura da clínica médica. Sistemas especialistas requerem que o conhecimento médico seja estruturado e modelado. Mas, a clínica médica não é uma ciência formal em que se pode pensar sempre por alguns princípios. A maior parte do conhecimento é geralmente experimental. Por esta razão, deve-se procurar a formalização da descrição de doenças e da linguagem médica e, paralelamente, estar cientes do nosso entendimento das doenças.

O segundo problema refere-se as metodologias disponíveis para projeto e construção de sistemas especialistas. É essencial uma visão mais sistemática desta tecnologia, definir quais as primitivas necessárias para o desenvolvimento destes sistemas em medicina, como combinar estas primitivas com estruturas de informação e, finalmente, que arquiteturas são necessárias para empregar estas estruturas.

A análise deste aspectos é, realmente, importante. A justificativa para sistemas especialistas na área médica deve ser comprovada. Geralmente, as principais justificativas referem-se a situações que colocam em risco a vida humana, onde existe o risco de perda do conhecimento na especialidade ou a não disponibilidade de especialistas para realizar a tarefa que o sistema resolveria [Waterman 86], [Kiper 92].

Encontramos, na literatura, muitas experiências de sistemas especialistas na área médica. Um dos mais conhecidos é o MYCIN, desenvolvido na Universidade de Stanford. O MYCIN auxilia médicos no diagnóstico e tratamento de pacientes com infecções bacteriológicas no sangue. Este sistema mostrou ter um desempenho comparável ao de especialistas em doenças infecciosas. Outros dois sistemas também bastante conhecidos são o INTERNIST-1, da Universidade de Pittsburgh, e o ONCOCIN, também da Universidade de Stanford. O INTERNIST-1 é um sistema de diagnóstico cuja base de conhecimento contém 400 doenças descritas pelos sintomas e sinais físicos e também mostrou ter um desempenho satisfatório quando comparado com médicos. O ONCOCIN é um sistema especialista que auxilia médicos no tratamento de pacientes com câncer através de protocolos de quimioterapia. Estes protocolos

comparam um regime de drogas com outro. A quimioterapia é caracterizada por diferentes tipos de tratamento com muitos tipos de drogas que variam todo tempo. Por causa da complexidade da base de conhecimentos os pesquisadores resolveram desenvolver editores de conhecimento nos quais os médicos podem informar estes protocolos. [Kane 88]

Estes sistemas podem ser considerados como marcos para pesquisas que continuam sendo realizadas em muitas universidades e hospitais. Estas pesquisas procuram experimentar o uso de sistemas especialistas em diferentes atividades (diagnóstico, interpretação, monitoração, etc) e em vários campos como, por exemplo: interpretação ou classificação de arritmias [Taddei 93], [Nadal 93], diagnóstico de infarto agudo do miocárdio através de ecocardiogramas e previsão do nível de vida [Micheli-Tzanakou 93], análise de eletrocardiograma [Nieberl 93], [Giakoumakis 93], diagnóstico de cardiopatia congênita [Leão 93], diagnóstico para pacientes que apresentam dores no peito [Assanelli 93], acompanhamento ou monitoramento de pacientes em unidades de tratamento intensivo [Artioli 93], [Fackler 93], diagnóstico de AIDS [Azevedo 94], diagnóstico de patologia orofaciais [Palombo 94], determinação e classificação de crises epilépticas [Shih-Min 94], tutores inteligentes [Khuwaja 93], [Sukthankar 93], [Direne 94], [Starita 94], planejamento e organização de exames médicos [Herren 93], etc.

Entretanto, embora um volume considerável de sistemas especialistas para a área médica já venham sendo desenvolvidos há muitos anos, alguns com bastante sucesso, várias questões no que se refere a seu processo de desenvolvimento e garantia da qualidade continuam em aberto e são objeto de pesquisas [Saranummi 91], [Kiper 92], [Sorare 93], [Kim 93].

Na próxima seção descrevemos nossa experiência, participando da equipe de desenvolvimento do projeto "Sistemas Especialistas em Cardiologia" (SEC) na Fundação Bahiana de Cardiologia. Serão destacados os aspectos relativos à garantia da qualidade do produto. Aspectos relacionados ao processo de desenvolvimento serão tratados apenas superficialmente e com o objetivo de permitir o entendimento dos procedimentos para controle da qualidade do produto, por serem objeto de outra na qual esta se integra [Werneck 95].

5.4 Desenvolvimento de um Sistema Especialista para Diagnóstico de Cardiopatia Isquêmica: Processo de Garantia da Qualidade

5.4.1 O Sistema SEC

Para melhor entendimento do contexto deste trabalho, iniciamos descrevendo brevemente os objetivos e principais características do sistema SEC (Sistema Especialista em Cardiologia).

A cardiopatia isquêmica é uma doença cardíaca decorrente da diminuição, significativa, do fluxo coronariano devido à formação de placas de gordura em determinados segmentos das artérias coronarianas e/ou redução do calibre por espasmos. A cardiopatia isquêmica pode se apresentar como angina estável, angina instável ou infarto agudo do miocárdio. Sendo as duas últimas definidas como eventos coronarianos agudos.

Duas características importantes nas ocorrências de atendimentos de pacientes com suspeita de cardiopatia isquêmica são: (*i*) a urgência na formulação do diagnóstico e nas medidas terapêuticas a serem tomadas, e, (*ii*) o fato de que o diagnóstico de eventos agudos da cardiopatia isquêmica está centrado na história clínica do paciente, no exame físico e no eletrocardiograma realizados [FBC 94a].

A primeira característica reflete a preocupação com o tempo nas situações de eventos coronarianos agudos. O tempo decorrido entre o diagnóstico preciso da dor e a tomada de uma decisão acertada é crucial, pois a maioria das complicações, inclusive o óbito, nos eventos coronarianos agudos ocorre nas primeiras horas. Existem, ainda, medidas terapêuticas que estão associadas ao tempo, como é o caso da trombólise. A administração de trombolíticos para desobstruir a artéria responsável pelo infarto, se feita nas primeiras horas, tem, aproximadamente, 80% de chance de sucesso. Este índice cai drasticamente até a 12ª hora e, daí em diante, há uma diminuição significativa do benefício trazido pelo seu uso. A própria indecisão do paciente quanto à sua ida a um posto de saúde gera uma perda de tempo inicial. Ao se definir que o paciente deve ser removido para um centro especializado, mais um gasto de tempo ocorre neste deslocamento. Ainda são gastos alguns minutos desde a sua chegada no centro especializado e o início da trombólise. Por isso, deve-se procurar encaminhar o paciente o mais rápido possível para locais com condições de realizar o tratamento adequado.

A segunda característica está relacionada com a história clínica, o exame físico e o eletrocardiograma como elementos básicos e essenciais para a elaboração do diagnóstico

com elevada chance de sucesso. Portanto, qualquer posto de saúde, mesmo que pequeno, poderia formular um diagnóstico de evento coronariano agudo. Porém, o que ocorre é que, muitas vezes, o diagnóstico não é feito pela falta de conhecimento especializado do médico que atende o paciente, isto é, muitas vezes o posto de saúde não dispõe de um cardiologista.

Neste contexto, um sistema especialista é sem dúvida de extrema validade, pois, a partir de um grupo de informações básicas iniciais, o sistema pode apoiar o médico não cardiologista na formulação do diagnóstico e no encaminhamento do paciente a uma unidade de referência especializada, quando este for necessário.

Considerando-se esta realidade, o SEC foi concebido com o objetivo de apoiar o médico, não cardiologista, no diagnóstico de eventos agudos da cardiopatia isquêmica [Finep/FBC 94]. Seus usuários serão, portanto, médicos não especialistas em cardiologia e o sistema será utilizado em unidades periféricas urbanas de atendimento, de natureza primária da rede pública de saúde. O SEC poderá, ainda, ser utilizado por estudantes de medicina para fins de aprendizado, sem no entanto ter as características de um tutorial.

Três cardiologistas e três engenheiros do conhecimento participaram do processo de aquisição do conhecimento. Utilizamos neste processo as seguintes técnicas: entrevistas, estudo de casos típicos, ordenação conceitual, tutorial e protocolo médico por telefone adaptada da técnica de análise de protocolo [Assis 92], [Gottgtroy 90] e [Scott 91]. O protocolo médico por telefone é uma simulação de um ligação telefônica entre um especialista e um médico. Podemos, assim, refinar nosso entendimento do raciocínio do especialista durante o diagnóstico pela observação de sua discussão com um médico residente em cardiologia. Para análise de conflito utilizamos uma adaptação da técnica de DELPHI [Turban 91]. Para isto as diferentes abordagens obtidas do conhecimento elicitado foram documentadas pelos engenheiros de software e apresentadas aos especialistas que avaliaram a documentação, individualmente, revendo suas opiniões. Caso persistissem as divergências, uma reunião era realizada para discutir os conflitos. A experiência obtida neste processo está relatada em [Rabelo 95].

O SEC é um sistema especialista baseado em regras com tratamento de incerteza. No final do processo é fornecido ao usuário-médico um diagnóstico e a sugestão da medida a ser tomada que se situa em quatro faixas (Figura 5.2) de acordo com o valor de certeza determinado.

FAIXA	CONCLUSÃO DO SEC
1 (0 a 4,99)	O paciente apresenta *muito baixa possibilidade* de estar com Evento Coronariano Agudo.
2 (5 a 5,99)	O paciente apresenta *baixa possibilidade* de estar com Evento Coronariano Agudo.
3 (6 a 6,99)	O paciente apresenta *possibilidade intermediária* de estar com Evento Coronariano Agudo.
4 (7 a 10)	O paciente apresenta *alta possibilidade* de estar com Evento Coronariano Agudo devendo ser imediatamente encaminhado para uma Unidade Coronariana.

Figura 5.2 - Faixas de conclusões determinadas pelo SEC

A área de Engenharia de Software da COPPE participa do projeto, desde seu início, em março de 1994, através de um convênio com a FBC, e é responsável pelos aspectos de definição e acompanhamento dos processos de desenvolvimento e garantia da qualidade.

5.4.2 Processo de Desenvolvimento do SEC

Desde o início do projeto houve a preocupação de que o sistema tivesse um processo sistemático de desenvolvimento e garantia da qualidade, utilizando técnicas modernas de Engenharia de Software. Para isso foi definido um processo de desenvolvimento adequado à construção de sistemas baseados em conhecimento de acordo com os princípios estabelecidos na norma ISO 9000-3 [ISO 9000-3]. Assim sendo, a primeira atividade no projeto foi a definição do *Processo de Desenvolvimento do Projeto Sistemas Especialistas em Cardiologia* [Werneck 94]. Este documento define o modelo de ciclo de vida adotado no projeto, os métodos e ferramentas para construção do produto, a documentação a ser produzida ao longo do desenvolvimento, os procedimentos e métodos para controle da qualidade e gerência do processo de desenvolvimento.

O ciclo de vida definido para o SEC [Werneck 95] está baseado no ciclo de vida de prototipagem evolutiva e a proposta de Hull [Hull 91] para desenvolvimento de sistemas

especialistas. Este ciclo de vida (Figura 5.3) é composto de três estágios: Análise da Viabilidade, Evolução e Maturidade.

Os estágios de desenvolvimento estão, por sua vez, divididos em etapas. A **Análise do Domínio do Problema** é uma etapa realizada somente uma vez no início do primeiro estágio (Análise da Viabilidade). Seu produto é a *Especificação do Domínio do Problema*, que abrange a definição do problema das áreas de conhecimento e do escopo do projeto. O modelo de ciclo de vida para cada estágio está organizado em até sete etapas que são aplicadas de forma incremental:

- *Planejamento do Projeto*, na qual é produzido inicialmente o *Plano do Projeto*. Em cada estágio este Plano é detalhado de acordo com o objetivo e características do estágio;
- *Análise do Conhecimento*, durante a qual os desenvolvedores elicitam o conhecimento e produzem o modelo conceitual para o estágio, descritos na *Especificação de Requisitos*;
- *Projeto*, onde o modelo conceitual é traduzido para o paradigma de Inteligência Artificial a ser utilizado. O produto desta fase é a *Especificação de Projeto*;
- *Construção*, onde o programa e a base de conhecimentos são construídos;
- *Avaliação do Produto*, onde os especialistas e outros avaliadores testam o sistema. O teste de aceitação determina se os requisitos descritos na Especificação de Requisitos foram alcançados;
- *Operação*, é a atividade que permite o sistema aceito ser utilizado;
- *Avaliação do Processo*, uma etapa fundamental, realizada no final de cada estágio, que é a base para revisões no processo de desenvolvimento e para o planejamento e realização do próximo estágio.

Para modelagem do sistema foi escolhido o método KADS (*Knowledge Acquisition and Design Structuring*) [Wielenga 88], [Hickman 89], [Schreiber 92]. Uma avaliação do uso do processo de desenvolvimento na construção da 1ª versão do SEC pode ser encontrada em [Rabelo 94a], [Rabelo 94b] e [Werneck 95].

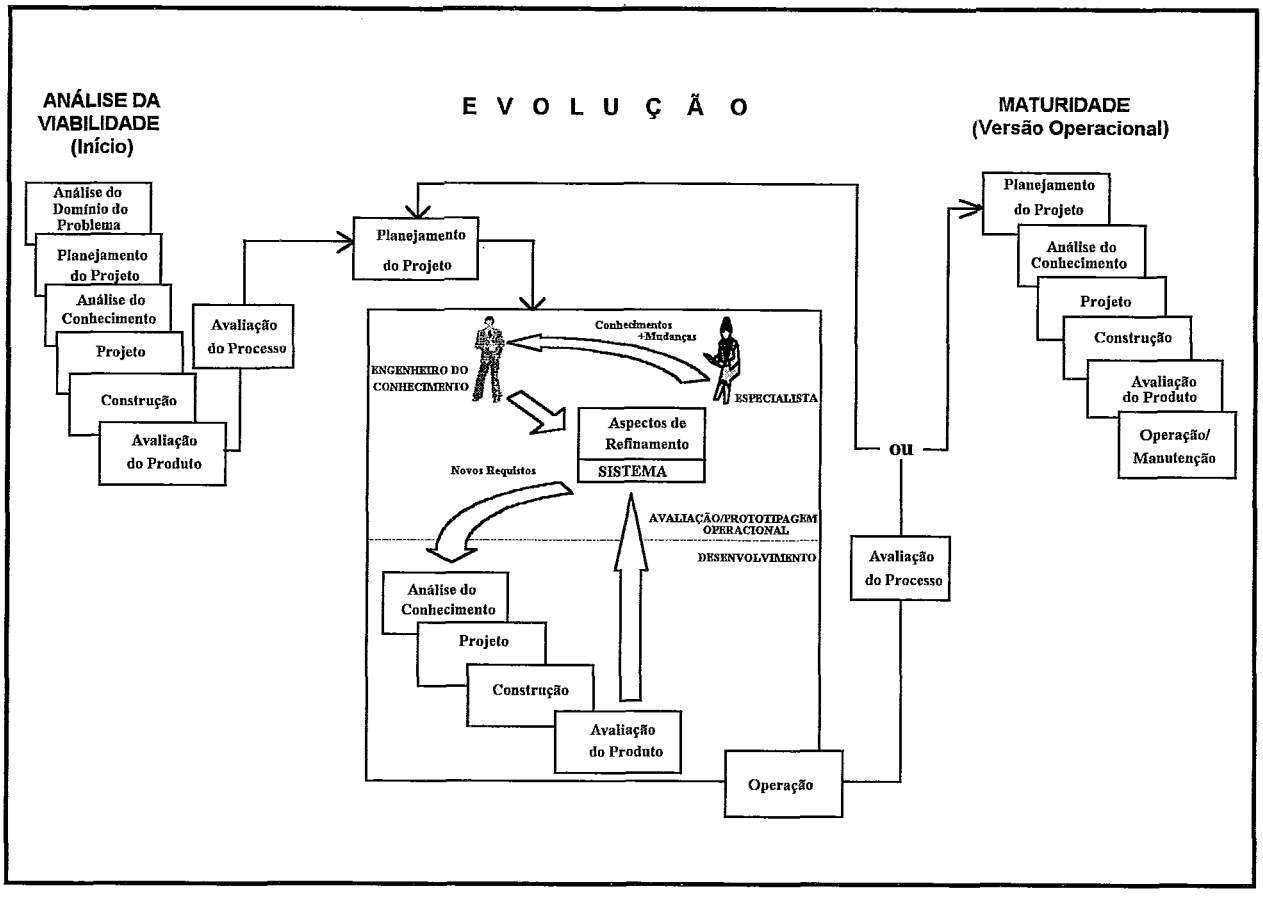

Figura 5.3 - Ciclo de Vida para Sistemas Especialistas [Werneck 95]

5.4.2.1 Planejamento da Qualidade do SEC

O Planejamento da Qualidade do projeto SEC foi realizado no início do 1º estágio de desenvolvimento (Análise da Viabilidade). Deve ser avaliado ao final de cada estágio na fase de avaliação do processo e, caso necessário, reformulado ao se planejar um novo estágio de desenvolvimento. O Plano de Controle da Qualidade faz parte, portanto, do Plano do Projeto.

Antes, porém, de se iniciar qualquer procedimento de controle da qualidade é fundamental que sejam identificados os requisitos de qualidade específicos de um projeto. Estes requisitos devem ser incorporados ao documento *Especificação de Requisitos de Software* e serão a base para o controle da qualidade.

Para estabelecermos os requisitos de qualidade do SEC, definimos um processo onde as características de qualidade para sistemas especialistas, definidas no capítulo anterior, foram divididas em dois conjuntos: atributos relacionados à qualidade externa do produto (percebida pelos cardiologistas) e atributos relacionados à qualidade interna (percebida pelos engenheiros de software).

Os atributos relacionados à qualidade externa do produto foram apresentados aos cardiologistas envolvidos no desenvolvimento do SEC e ao coordenador do projeto, que foram solicitados a identificar, entre estes, os atributos imprescindíveis, os desejáveis e os considerados sem importância para o sistema. Os atributos identificados como imprescindíveis e desejáveis pelos cardiologistas, acrescidos dos atributos de qualidade interna do produto, identificados pelos engenheiros de software, foram, então, definidos como os *Requisitos de Qualidade do SEC* [Oliveira 94b], [FBC 94a]. A Figura 5.4 mostra, como exemplo, parte do formulário submetido aos cardiologistas. As figuras 5.5 e 5.6 mostram os requisitos de qualidade do SEC, respeitando a importância estabelecida pelos cardiologistas e engenheiros de software.

A avaliação da qualidade do SEC foi planejada para ser realizada ao longo de todo o processo de desenvolvimento. Durante as diversas etapas do projeto foram planejados dois tipos de avaliações: avaliações intermediárias e avaliações finais de produtos referentes às fases do processo de desenvolvimento.

Avaliações intermediárias devem ser realizadas ao se dar por concluida uma tarefa cujo resultado vá impactar fortemente na próxima tarefa. Esta avaliação tem como objetivo ter-se uma aprovação, conforme pertinente em cada situação, dos especialistas, usuários ou da equipe técnica sobre a adequação e completeza do produto da tarefa.

Avaliações finais de produtos são sempre realizadas ao se dar por concluídas as tarefas de uma etapa e antes de se passar à próxima etapa. Esta avaliação tem como objetivo obter-se a aprovação do coordenador do projeto e dos especialistas para o produto da etapa.

As avaliações são realizadas através da técnica de *inspeção por fases* [Knight 93]. Escolhemos esta técnica para realização das avaliações já que diferentes requisitos de qualidade deveriam ser analisados em um mesmo produto, por avaliadores com diferentes perfis e nem sempre com possibilidade de estarem presentes às reuniões de inspeção. Por exemplo, a modelagem do sistema foi planejada para ser analisada quanto à correção da representação por especialistas em engenharia de software e no uso do método KADS. A correção da especificação foi planejada para ser avalidada pelos especialistas em cardiologa, os únicos capazes de realizá-la. O Anexo I mostra, como exemplo, os dois formulários de avaliação.

ATRIBUTO	DESCRIÇÃO	IMPRES-CINDÍVEL	DESEJÁVEL	SEM IMPORTÂNCIA	IMPORTANTE 1ª VERSÃO
Utilidade	característica da utilização do sistema ter consequências significativamente úteis				
Amenidade de uso	característica do sistema apresentar os resultados de forma clara, ser capaz de disponibilizar auxílio de forma rápida e apresentar estabilidade no uso				
Equivalência a especialista	característica do sistema ter um desempenho com confiabilidade equivalente a de um especialista, tanto em termos de escopo do problema como da solução gerada				
Segurança	características do sistema ser confiável sob o ponto de vista de seu uso, sem risco de provocar danos materiais e/ou de vidas humanas, protegendo-se de violações e garantindo a privacidade				
...					

Figura 5.4 - Formulário para Identificação dos Requisitos de Qualidade do SEC

IMPRESCINDÍVEL	DESEJÁVEL
Grau de Explicação	Adequação da metodologia
Suporte	Inferência
Adequação do ambiente de programação	Estilo
Codificabilidade	Uniformidade do grau de abstração
Clareza	Rastreabilidade
Concisão	Adaptabilidade
Modularidade	
Correção da representação	
Uniformidade de terminologia	
Completude da documentação	
Disponibilidade da documentação	
Estrutura	

Figura 5.5 - Requisitos de Qualidade Identificados pelos Engenheiros de Software

IMPRESCINDÍVEL	DESEJÁVEL
Modificabilidade	Compatibilidade com outros sistemas
Evolutibilidade	Adaptabilidade
Facilidade de aprendizado	Eficiência de armazenamento
Atualização de aprendizado	Lucratividade
Amenidade de uso	Competitividade
Interatividade	
Facilidade de interfaceamento	
Independência do ambiente	
Oportunidade	
Eficiência de execução	
Benefício social	
Benefício no trabalho do usuário	
Relevância	
Utilidade	
Justificabilidade	
Possibilidade	
Adequabilidade	
Viabilidade econômica	
Viabilidade financeira	
Viabilidade tecnológica	
Viabilidade de mão de obra	
Viabilidade de cronograma	
Viabilidade social	
Correção da especificação	
Confiabilidade da implementação	
Completude	
Exaustividade	
Não redundância	
Especificidade	
Consistência	
Equivalência a especialista	
Robustez	
Segurança	

Figura 5.6- Requisitos de Qualidade Identificados pelos Cardiologistas

As diferenças entre software convencional e sistemas especialistas determinam peculiaridades na validação destes sistemas. Como nenhum método pode assegurar a correção total de um sistema, decidimos usar uma combinação de vários métodos de validação de forma a ter uma evidência complementar da confiabilidade do sistema [O'Keefe 87], [Meseguer 92], [Meseguer 93] e [O'Leary 87]. Assim sendo, o planejamento da qualidade do SEC inclui a realização de testes sistemáticos, com casos de teste previamente definidos e avaliados quanto à sua adequação e correção do resultado esperado. Para o 1º estágio de desenvolvimento foram previstas as técnicas de

validação face a face, onde os especialistas deveriam comparar o desempenho do sistema com seu próprio desempenho e a técnica de *análise de sensibilidade*, onde os especialistas sistemáticamente mudam o valor das váriáveis de entrada, observando seu efeito no desempenho do sistema.

A partir destas decisões sobre os requisitos de qualidade, métodos e técnicas a serem utilizados para garantia da qualidade do produto no projeto SEC, foi elaborado o *Plano para Controle da Qualidade* (Anexo II), que faz parte do documento *Plano do Projeto Sistemas Especialistas em Cardiologia* [FBC 94b].

5.4.3 O Processo de Controle da Qualidade

Nesta seção descrevemos como foi realizado o controle da qualidade durante o 1º estágio de desenvolvimento do SEC e os resultados obtidos com este processo.

Ao longo do desenvolvimento da versão 1.0, o sistema passou por um processo rigoroso de garantia da qualidade onde foram realizadas várias avaliações em marcos e pontos de controle pré-estabelecidos no *Plano de Controle da Qualidade* (Anexo II). Estas avaliações foram realizadas através de inspeções, segundo os requisitos de qualidade estabelecidos na *Especificação de Requisitos do Sistema* e utilizando-se critérios específicos para a avaliação de cada produto intermediário e dos produtos finais das etapas de desenvolvimento, conforme descrevemos na seção anterior. Para facilitar a avaliação foram definidos formulários, considerando-se o perfil de cada avaliador (coordenador do projeto, especialistas, avaliadores externos e engenheiros de software) e o produto específico a ser avaliado. O Anexo III contém o conjunto dos formulários elaborados para a avaliação do produto final da versão 1.0. Para esta versão foram considerados, apenas, parte dos atributos de qualidade desejados para o produto. Esta seleção foi feita pelos cardiologistas (Figura 5.4) ao estabelecerem os requisitos de qualidade do sistema.

Na figura 5.7 mostramos que atributos de qualidade foram avaliados em cada produto, intermediário ou final, ao longo do desenvolvimento da versão 1.0.

Também foram utilizadas reuniões de *walkthrough* nas avaliações realizadas com os cardiologistas pois verificamos situações onde a formalidade imposta pelas reuniões de inspeção não era adequada.

A validação do sistema, através de testes, foi feita inicialmente usando *validação face a face*. O passo seguinte no processo de validação foi realizar *análise de sensibilidade*. Estes dois métodos de validação foram apoiados por prototipagem operacional [Davis 92] de forma a acelerar o processo de validação da versão 1.0

PRODUTO	ATRIBUTOS
Especificação do Domínio do Problema	
Conhecimento Geral	Clareza, concisão, uniformidade da terminologia, uniformidade do grau de abstração, completude da documentação, justificabilidade, possibilidade, adequabilidade, correção da especificação, especificidade, não redundância, consistência
Documento final	Clareza, concisão, uniformidade da terminologia, uniformidade do grau de abstração, completude da documentação, disponibilidade da documentação, rastreabilidade, relevância, utilidade, justificabilidade, possibilidade, adequabilidade, correção da especificação especificidade, não redundância, consistência
Plano do Projeto	
Documento Final	Clareza, completude da documentação, disponibilidade da documentação, rastreabilidade, viabilidade econômica, viabilidade financeira, viabilidade de tecnologia, viabilidade de mão de obra, viabilidade de cronograma, viabilidade social,
Especificação de Requisitos	
Objetivo, requisitos do sistema	Clareza, concisão, uniformidade de terminologia, uniformidade do grau de abstração, completude da documentação, correção da especificação, especificidade, não redundância, consistência
Modelagem do sistema	Clareza, uniformidade de terminologia, completude da documentação, uniformidade do grau de abstração, correção da especificação, correção da representação, especificidade, não redundância, consistência
Documento Final	Clareza, concisão, uniformidade de terminologia, uniformidade do grau de abstração, completude da documentação, rastreabilidade, correção da especificação, especificidade, não redundância, consistência, disponibilidade da documentação

Figura 5.7 - Atributos de Qualidade Avaliados em cada Produto ao longo do Desenvolvimento

PRODUTO	ATRIBUTOS
Especificação de Projeto	
Base de Conhecimento	Clareza, uniformidade de terminologia, completude da documentação, estrutura, uniformidade do grau de abstração, codificabilidade, viabilidade tecnológica, correção da especificação, correção da representação, especificidade, não redundância, consistência
Documento Final	Clareza, uniformidade de terminologia, uniformidade do grau de abstração, completude da documentação, estrutura, rastreabilidade, viabilidade tecnológica, correção da especificação, correção da representação, especificidade, não redundância, consistência, codificabilidade
Produto Final	
Programas, Documentação do Projeto	Facilidade de aprendizado, amenidade de uso, interatividade, oportunidade, eficiência da execução, grau de explicação, adequabilidade, confiabilidade da implementação, especificidade, não redundância, consistência, uniformidade de terminologia, uniformidade no grau de abstração, completude da documentação, disponibilidade da documentação, rastreabilidade, adequação da metodologia, adequação do ambiente de programação, codificabilidade, clareza, concisão, modularidade, correção da representação, estrutura

Figura 5.7 (cont.) - Atributos de Qualidade Avaliados em cada Produto ao longo do Desenvolvimento

Após a aprovação da versão 1.0, em setembro de 1994, o sistema passou por uma série de refinamentos durante os meses de outubro, novembro e dezembro de 1994, que deram origem às versões 1.1, 1.2 e 1.3.

Os casos de teste utilizados para a validação das versões 1.0, 1.1, 1.2 e 1.3 foram elaborados pelos especialistas envolvidos no projeto e por médicos residentes da FBC. A validação foi feita utilizando análise de sensibilidade. Todos os casos de teste foram objeto de reuniões de inspeção onde se avaliou sua adequação e a correção do resultado esperado que deveria se situar numa das quatro faixas definidas na seção 5.4.1. O Anexo IV contém um exemplo de formulário para elaboração de casos de teste do SEC, visando analisar a sensibilidade do sistema. Na figura 5.8 mostramos um exemplo de caso de teste básico com as variações para análise de sensibilidade.

PROJETO SEC - Casos de Teste

Responsável pelo Caso de Teste: XXXXX
Data: XX/XX/XX

CASO BÁSICO (Caso 18)	VARIAÇÃO 1 (Caso 19)	VARIAÇÃO 2 (Caso 20)		
Sexo masculino				
48 anos				
dor precordial				
em aperto ou opressão				
dor severa		dor leve		
10 horas (600 min)		2 horas (120 min)		
palpitações frequentes		sem palpitação		
tabagista		sem tabagismo		
stress		sem stress		
dor a palpação na região anterior do torax	sem dor a palpação	exame físico normal		
ECG normal	T negativa	ECG normal		
	+ palidez			
	RESULTADO ESPERADO			
1	4	3		

Figura 5.8 - Formulário para Registro de Caso de Teste para Análise de Sensibilidade

Foram realizados 88 casos de teste. As figuras 5.9, 5.10, 5.11 e 5.12 mostram o comportamento do sistema para os casos com resultado esperado em cada uma das quatro faixas consideradas. Nestes gráficos, o eixo x contém as diferentes versões e o eixo y a possibilidade de evento coronariano agudo, determinada pelo sistema, numa escala de 0 a 10. A partir destas figuras pode-se observar que:

- nos casos com resultado esperado na faixa 1, o sistema inicialmente tendia a valorizar excessivamente as características apresentadas pelo paciente e, progressivamente, esta valorização foi sendo ajustada de forma que na versão 1.3, para maioria dos casos, obteve-se o valor esperado;
- nos casos com resultado esperado nas faixas 2 e 3, teve-se um número de casos de teste muito pouco significativo para se fazer qualquer afirmação, e,
- nos casos com resultado esperado na faixa 4, é possível observar um comportamento, praticamente, constante do sistema nas quatro versões.

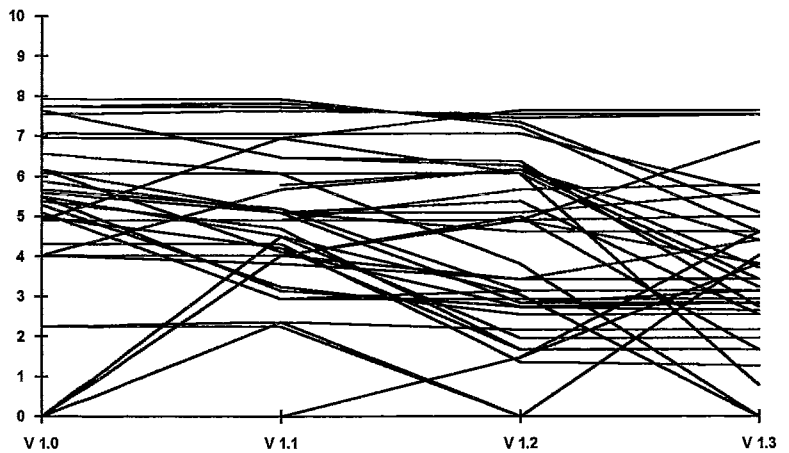

Figura 5.9 - Casos com resultado esperado na faixa 1 (0 a 4,99)

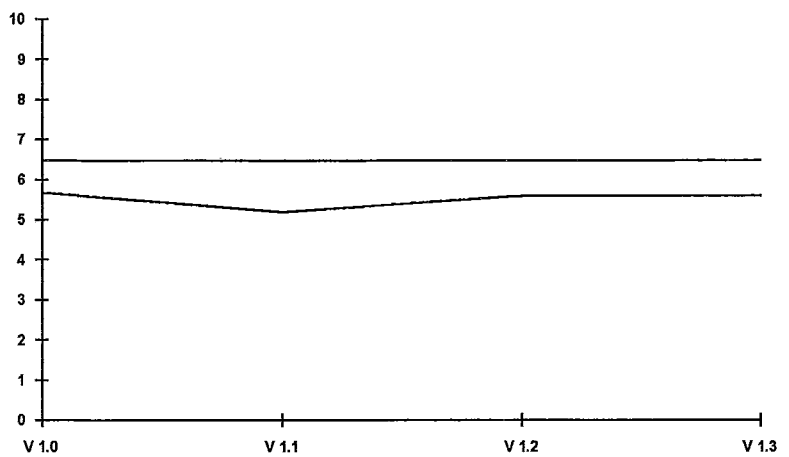

Figura 5.10 - Casos com resultado esperado na faixa 2 (5 a 5,99)

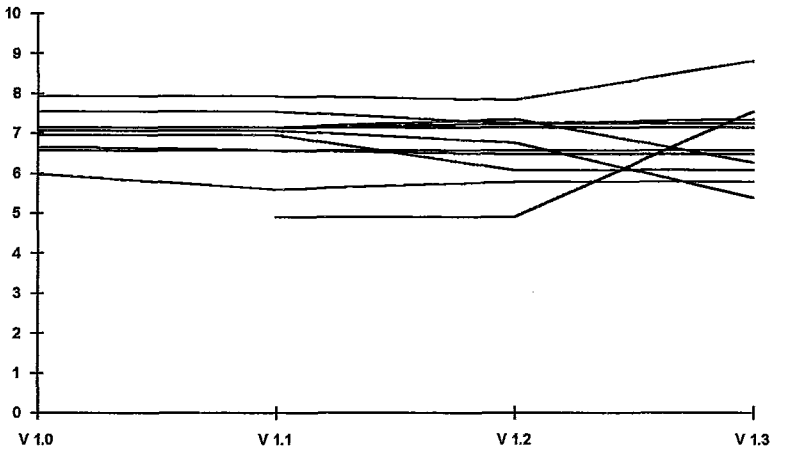

Figura 5.11 - Casos com resultado esperado na faixa 3 (6 a 6,99)

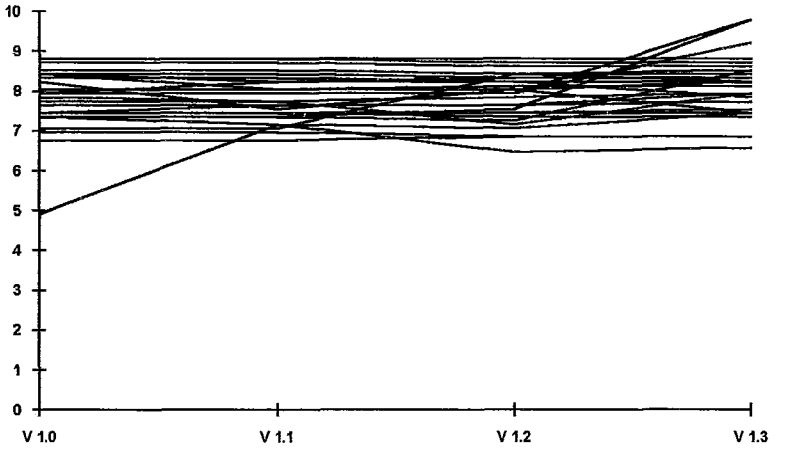

Figura 5.12 - Casos com resultado esperado na faixa 4 (7 a 10)

Finalmente, foi feita uma *análise estatística* dos resultados obtidos com os 88 casos de teste, nas quatro versões. O objetivo desta análise foi verificar o grau de confiança (acurácia) no diagnóstico fornecido pelo sistema em cada versão e identificar se houve, como desejado, um ganho de confiabilidade de uma versão para a outra. Foram estabelecidas duas hipóteses:

- *Hipótese 1- O Sistema é acurado.*

O objetivo é verificar o grau de confiança na conclusão diagnóstica fornecida pelo sistema. O que se deseja é avaliar em que medida o sistema fornece resultados iguais ao esperado, isto é, à conclusão diagnóstica fornecida por um especialista.

Para verificar esta hipótese foi feito o teste de Wilcoxon comparando os resultados obtidos em cada versão com os resultados esperados, por tratar-se de populações dependentes onde não se pode afirmar se existe distribuição normal. Foi usado o pacote estatístico SPSS versão 5.0 para Windows.

Decidiu-se aceitar a hipótese 1 se $p > 0,05$ (0,05 é o nível de significância normalmente usado em medicina [Bevilacqua 94]). Os resultados obtidos estão na Figura 5.13.

VERSÃO	*p*
V 1.0	0,0000
V 1.1	0,0001
V 1.2	0,0002
V 1.3	0,0112

Figura 5.13 - Resultados obtidos com o teste de Wilcoxon para a hipótese 1

Analisando estes resultados a hipótese 1 é rejeitada, o que significa que o sistema, ainda não atingiu o grau de acurácia desejado. Entretanto, pode-se verificar que este grau cresce de uma versão para a outra evidenciando uma tendência na direção desejada.

♦ *Hipótese 2 - As versões aumentaram o grau de confiança na acurácia do sistema.*

O objetivo é verificar se há um crescimento significativo na acurácia do sistema de uma versão para outra, o que significa verificar se o processo de refinamento utilizado foi eficaz.

Espera-se que haja um aumento mais significativo de acurácia da versão inicial (versão 1.0) para a versão final (versão 1.3) e um pequeno aumento entre as versões intermediárias. Foi estabelecido como aumento significativo da versão 1.0 para versão 1.3 se $p < 0,01$ e entre versões se $p < 0,05$.

Para verificar esta hipótese também foi feito o teste de Wilcoxon, utilizando o SPSS e comparando os resultados obtidos entre as versões (Figura 5.14).

VERSÕES COMPARADAS	*p*
V 1.0 → V 1.1	0.0442
V 1.0 → V 1.2	0,0347
V 1.0 → V 1.3	0,0005
V 1.1 → V 1.2	0,6496
V 1.1 → V 1.3	0,0115
V 1.2 → V 1.3	0,0245

Figura 5.14 - Resultados obtidos com o teste de Wilcoxon para a hipótese 2

Pode-se, então afirmar que houve aumento significativo em todos os casos menos no que se refere a comparação da versão 1.1 para a versão 1.2, o que mostra a eficácia do processo de refinamento das versões.

Analisou-se ainda, o comportamento do sistema com relação a cada faixa de resultado esperado.

Para os casos com resultado esperado na faixa 2, o número de casos de teste é insuficiente para se poder tirar qualquer conclusão sobre o comportamento do sistema.

Para os casos com resultado esperado nas faixas 1 , 3 e 4 (Figuras 5.15, 5.16 e 5.17) observamos que no início o sistema era muito sensível e muito pouco específico, ou seja, casos em que o paciente não apresentava evento coronariano agudo e o sistema consierou falso positivo (foram considerados ter alguma possibilidade de apresentar evento coronariano agudo) e, até mesmo, encaminhar o paciente para unidade coronariana. A medida que o sistema foi sendo refinado percebe-se uma tendência para se conseguir especificidade, sem perder a sensibilidade para os casos em que o paciente apresenta, realmente, evento coronariano agudo e que deve ser encaminhado para a unidade coronariana.

VERSÃO	RESULTADO OBSERVADO			
	FAIXA 1 (0 a 4,99)	FAIXA 2 (5 a 5,99)	FAIXA 3 (6 a 6,99)	FAIXA 4 (7 a 10)
V 1.0	17	12	6	9
V 1.1	24	9	6	5
V 1.2	27	4	7	6
V 1.3	36	4	1	3

Figura 5.15 - Número de casos com resultado observado em cada faixa para os casos de teste com resultado esperado na faixa 1

VERSÃO	RESULTADO OBSERVADO			
	FAIXA 1 (0 a 4,99)	FAIXA 2 (5 a 5,99)	FAIXA 3 (6 a 6,99)	FAIXA 4 (7 a 10)
V 1.0	1	1	4	8
V 1.1	1	1	4	8
V 1.2	1	1	6	6
V 1.3	0	2	6	6

Figura 5.16 - Número de casos com resultado observado em cada faixa para os casos de teste com resultado esperado na faixa 3

VERSÃO	RESULTADO OBSERVADO			
	FAIXA 1 (0 a 4,99)	FAIXA 2 (5 a 5,99)	FAIXA 3 (6 a 6,99)	FAIXA 4 (7 a 10)
V 1.0	0	0	2	26
V 1.1	0	0	2	26
V 1.2	0	0	2	26
V 1.3	0	0	2	26

Figura 5.17 - Número de casos com resultado observado em cada faixa para os casos de teste com resultado esperado na faixa 4

Considerando que sistemas especialistas da área médica devem se preocupar em não oferecer riscos ao paciente, podemos dizer que o SEC, até o momento, é confiável sob este ponto de vista, pois nenhum paciente é liberado quando apresenta alguma possibilidade de estar com evento coronariano agudo.

O Anexo IV contém as listagens dos resultados gerados pelo SPSS, relativas aos testes realizados para a análise quantitativa.

A versão 1.3 é um sistema especialista com 152 regras e 36 variáveis, para a qual obtemos 69 diagnósticos corretos (isto é, exatamente dentro da faixa esperada) dos 88 casos de teste submetidos ao sistema.

A avaliação do 1º Estágio quanto aos aspectos de planejamento e controle da qualide nos levaram a planejar a inclusão no 2º Estágio, de validação através de *Teste em campo* [O'Keefe 87] e *Teste de Turing* [O'Keefe 87], [Chandrasekaran 83], [O'Leary 87], [Harrison 94].

O teste de campo será feito em três estágios. Inicialmente será realizado, apenas, no Pronto Atendimento da UCCV/FBC por ser este um ambiente totalmente sob o controle da Instituição. Numa segunda etapa o sistema será instalado em três hospitais da Rede Estadual de Saúde em Salvador. Finalmente, será instalado em três postos de saúde, também de Salvador. O Anexo V contém os formulários para a elaboração de casos para o Teste de Turing e registro das ocorrências nos testes em campo.

5.5 Conclusão

Neste capítulo apresentamos algumas questões atualmente relacionadas ao desenvolvimento e avaliação de software médico e descrevemos, com detalhes o processo de avaliação da qualidade do projeto Sistemas Especialistas em Cardiologia (SEC) da Fundação Bahiana de Cardiologia. A participação no processo de desenvolvimento e avaliação da qualidade deste projeto, nos permitiu aprofundar a questão proposta nesta tese e avaliar a qualidade de um sistema especialista segundo as características definidas no capítulo anterior.

A experiência nos mostrou a importância de se ter um Plano de Controle da Qualidade sistemático, escrito, bem definido e aprovado pela coordenação do projeto. O 1º estágio do SEC foi realizado absolutamente dentro do cronograma previsto e seus resultados surpreenderam os cardiologistas quanto à qualidade do sistema.

Certamente vários melhoramentos ainda podem ser feitos no processo de controle da qualidade, especialmente no que se refere à definição mais precisa de critérios, métricas e procedimentos quantitativos de avaliação. Sabemos, porém, que este é um processo contínuo e que só através de experiências como a realizada na FBC podemos conhecer melhor o problema e, assim, buscar melhores procedimentos de garantia da qualidade.

Capítulo 6
Conclusão

Esta tese abordou questões relacionadas à avaliação da qualidade de sistemas especialistas. Foi realizado um estudo referente ao estado da arte destes sistemas, onde se constatou a necessidade de uma melhor definição dos atributos que contribuam para a sua qualidade. Foi, então, proposto um conjunto de atributos de qualidade a serem considerados na avaliação dos diferentes produtos gerados ao longo do processo de desenvolvimento (especificações e código) e do produto final.

Partindo deste conjunto de atributos e do estudo da literatura sobre verificação e validação de software e, especificamente, sistemas especialistas, foram definidos os procedimentos para avaliação da qualidade de um sistema especialista para diagnóstico em Cardiologia, o sistema SEC (Sistema Especialista em Cardiologia), em desenvolvimento na Fundação Bahiana de Cardiologia (FBC).

O processo de avaliação da qualidade do SEC contemplou todos os produtos intermediários gerados ao longo de seu desenvolvimento, o que significou estabelecer que atributos seriam avaliados em cada momento e os processos de avaliação específicos. A avaliação do SEC utilizou inspeções e técnicas de validação qualitativa como análise de sensibilidade e validação face a face propostas na literatura.

São, portanto, contribuições deste trabalho:

- a definição de um conjunto de atributos a serem considerados na avaliação da qualidade de sistemas especialistas;
- a organização destes atributos no modelo proposto por Rocha;
- a definição de um processo sistemático de avaliação da qualidade ao longo do desenvolvimento de um sistema especialista;
- o relato de uma experiência concreta de avaliação de um sistema especialista desde as primeiras fases de seu desenvolvimento até o produto final.

Muito, entretanto, ainda pode ser feito. No que se refere à avaliação do SEC, outras técnicas serão utilizadas na segunda versão do sistema. Está planejado o uso do Teste de Turing e validação em campo em três hospitais e três postos de saúde de Salvador, bem

como uma avaliação da base de conhecimento no que diz respeito à sua completude e consistência.

Esta avaliação da base de conhecimentos não foi realizada devido ao fato, de termos realizado apenas o primeiro estágio de desenvolvimento do SEC. Este estágio teve como objetivo analisar a viabilidade de construção de um sistema especialista para a área e o assunto considerado, experimentar o uso de um processo de desenvolvimento específico e avaliar qual o melhor paradigma de Inteligência Artificial para representação do problema. Dessa forma, não foram utilizados o ambiente de programação definitivo nem o melhor paradigma de representação. Para o segundo estágio de desenvolvimento, em que já temos todos estes aspectos definidos, está planejada uma validação da base de conhecimentos de acordo com o que propõe a literatura.

No que se refere à pesquisa na área de qualidade de software, outros trabalhos devem ser realizados. Sentimos a necessidade de definição métricas mais precisas e objetivas para a avaliação da qualidade de sistemas especialistas. Esta área, ainda é muito pouco explorada na literatura devendo ser melhor pesquisada e experimentada.

Estamos convencidos de que para o desenvolvimento de trabalhos como o desta tese e para o avanço do conhecimento na área são necessárias experiências como a que foi realizada na FBC. A definição de procedimentos adequados para avaliação da qualidade de software não se faz sem que, a partir de uma primeira definição, estes procedimentos sejam experimentados, avaliados e submetidos a revisão. A continuação do projeto nos dará, portanto, a possibilidade de experimentar novos procedimentos de avaliação e refinar o processo de medição da qualidade.

Outras possibilidades de trabalho referem-se ao desenvolvimento de ferramentas específicas, que apoiem o processo de verificação e validação de sistemas especialistas.

Referências Bibliográficas

Ackerman 89 — Ackerman, A.F., Buchwald, L.S. e Lewski, F.H.; "Software Inspections: An Effective Verification Process"; **IEEE Software**; maio 1989

Aguiar 92 — Aguiar T.C.; **Um Sistema Especialista de Suporte à Decisão para Planejamento de Ambientes de Desenvolvimento de Software**; Tese de Doutorado, COPPE/UFRJ; 1992

Andrade 91 — Andrade, C.J. de; **ANAFOR: um analisador de Programas FORTRAN**; Tese de Mestrado, COPPE/UFRJ; abril 1991

ANSI/IEEE Std 730- 93 — ANSI/IEEE Std-730; "Standard for Software Quality Assurance Plans (ANSI)"; **IEEE Standards Collection - Software Engineering**; 1993

Arthur 93 — Arthur, L.J; **Improving Software Quality - An Insider's Guide to TQM**; Wiley Professional Computing; 1993

Artioli 93 — Artioli, E. e Ursino, M.; "Integration of Qualitative Reasoning to Support Medical Decision in Cardiac Intensive Care Units"; **Computers in Cardiology**; IEEE Computer Society Press; Londres, Inglaterra; setembro 1993

Assanelli 93 — Assanelli, D., Cazzamalli, L., Stambini, M. e Poeta, M.L; "Correct Diagnosis of Chest Pain by an Integrated Expert System"; **Computers in Cardiology**; IEEE Computer Society Press; Londres, Inglaterra; setembro 1993

Assis 92 — Assis, S.G. de; **Aquisição de Conhecimento para Sistemas Baseados em Conhecimento de Software**; Tese de Mestrado, COPPE/UFRJ; 1992

Azevedo 94 — Azevedo, L.O e Castilho, E.A.; "AIDS: sistema especialista para auxílio ao diagnóstico de AID segundo os critérios do CDC, da OPAS e do Ministério da Saúde"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Bahia 92 — Bahia, A. de Sá; **Avaliação da Complexidade de Software Científico**; Tese de Mestrado, COPPE/UFRJ, maio 1992

Banta 92 — Banta, H.D; "Quality Assurance Issues and PACS"; **Internacional Journal Biomedicine Computer**; Elsevier Scientific Publishers Irelan Ltd.; 1992

Basili 91 Basili, V. e Musa, J.; "The Future Engineering of Software: A Management Perspective"; **IEEE Computer**; setembro 1991

Bazzana 93 Bazzana, G., Andersen, O. e Jokela, T.; "ISO 9126 and ISO 9000: Friend or Froes?"; **Software Engineering Standards Symposium**; Brighton, Inglaterra; agosto 1993

Belchior 92a Belchior, A.D; **Características de Qualidade de Programas**; Publicações Técnicas, COPPE/UFRJ; julho 1992

Belchior 92b Belchior, A.D; **Avaliação da Qualidade de Software Financeiro**; Tese de Mestrado, COPPE/UFRJ; 1992

Belchior 94 Belchior, A.D.; "Meta-Avaliador de Especificações"; **Workshop Qualidade de Software**; Curitiba, PR; outubro 1994

Belden 93 Belden, T. et al; "Developing an Advanced Tool for the Clinician: Using Industrial Design and Interface Design Together to Bribg Technology into the Hand of the User"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Bench-Capon 93 Bench-Capon, T. et al; "Two Aspects of The Validation and Verification of Knowledge-Based Systems"; **IEEE Expert**; junho 1993

Bevilacqua 94 Bevilacqua, R.G.; **Estatística - Um roteiro para Médicos**; FBC; novembro 1994

Boegh 93 Boegh, J., Hausen, H-L e Welzel, D.; "A Practioners Guide to Evaluation of Software"; **IEEE Software**; 1993

Boehm 78 Boehm, B.; **Characteristcs of Software Quality**; North-Holland; 1978

Boehm 80 B.; **Software Engineering - AS IT IS**; Academic Press; 1980

Botten 89 Botten, N., Kusiac, A. e Raz, T.; "Knowledge-bases: Integration, verification and partitioning"; **European Journal of Operations Research 42**; 1989

Bundy 87 Bundy, A.; "How to Improve the Reliability of Expert Systems"; **Seventh Annual Conference of the Pontish Computer Society Specialist Group on Expert Systems**; dezembro 1987

Campos 94a Campos, G.H.B. de; **Metodologia para avaliação da qualidade de software educacional: Diretrizes para desenvolvedores e usuários**; Tese de Doutorado, COPPE/UFRJ; novembro 1994

Campos 94b Campos, F.C; **Hipermídia na Educação: Paradigmas e Avaliação da Qualidade**, Tese de Mestado; agosto 1994

Cardoso 87 Cardoso, C.C. de; **Árvore DSAE: Um Método para Controle da Qualidade de Software**, Tese de Mestrado, IME; fevereiro 1987

Carvalho 89 Carvalho, L.A.V.; **Síntese de Redes Neuronais com Aplicações à Representação do Conhecimento**, Tese de Doutorado, COPPE/UFRJ; 1989

Chandrasekaran 83 Chandrasekaran, B.; "On Evaluation AI Systems for Meducal Diagnosis"; **AI Magazine**; 1983

Chao 93 Chao, S-C. et al; "The MEDAS Local Area Network"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Chen 93 Chen, K-L et al; "A Comparison Among Different Front-End Input Versions of a Medical Record System"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Clunie 87 Clunie, C. E.; **Verificação e Validação de Software na fase de Especificação de Requisitos**; Tese de Mestrado, COPPE/UFRJ; fevereiro 1987.

Clunie 94 Clunie, C.E., Rocha, A.R.C.da e Werner, C.M.L.; "Critérios de Qualidade para um Modelo Orientado a Objetos"; **Workshop Qualidade de Software**; Curitiba, PR; outubro 1994

Collins 94 Collins, W.R., Miller, K.W., Spielman, B.J. e Wherry, P.; "How good is good enough? An Ethical Analysis of Software Construction and Use"; **Communications of the ACM**; janeiro 1994

Commerlato 94 Commerlato, C.; Xexeo, G.B.; Rocha, A.R.C. da "Avaliação da Reutilizabilidade de Componentes de Software"; **VIII Simpósio Brasileiro de Engenharia de Software**; Curitiba, PR; outubro 1994

Conrath 91 Conrath, D.W. e Sharma, R.S.; "Evaluating Expert Systems Using A Multiple-Criteria, Multiple-Stakeholder Approach"; **IEEE Expert**; 1991

Coutinho 94 Coutinho, M. et al; "Sistema de Informação em cardiologia"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Coward 88 Coward, P.D.; "A Review of Software Testing"; **Information and Software Tecnology**; vol 30, nº 3; abril 1988

Cragun 87 Cragun, B.J. e Steudel, H.J.; "A decision-table-based processor for checking completeness and consistency in rule base expert systems"; **Man-Machine Studies**; 1987

Davis 92 Davis, A.M.; "Operational Prototyping: a new development approach"; **IEEE Software**; 1992

Deon 94 Deon, R. e Notare, M.S.M.A.; "Sistema de vigilância de controle de infecções hospitalares"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Dhawan 93 Dhawan, A.P. et al; "Image Analisys and 3-D Visualization of Intracerebral Brain Hemorrhage"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Dijkstra 76 Dijkstra, E.; **A Discipline of Programming**; Prentice-Hall; 1976

Direne 94 Direne, A.I. et al; "Tutores automáticos para o ensino de radiologia"; **IV congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Duarte 91 Duarte, M.A.G.; **Um Sistema para Representação do Conhecimento de Métodos de Desenvolvimento de Software**; Tese de Mestrado, COPPE/UFRJ; 1991

Dyer 87 Dyer, M.; "A Formal Approach to Software Error Removal"; **Journal System Software** 7; 1987

Elahi 93 Elahi, B.J.; "Safety and Hazard Analysis for Software Controlled Medical Devices"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Eureka 93 Eureka, W.E..; "FDA and the Regulation on Medical Software"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Fackler 93 Fackler, J. C. e Kohane, I.S.; "Integration on Intermittent Clinical Data with Continuos Data from Bedside Monitors"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Fagan 76 Fagan, M. E.; "Design and Code Inspection to Reduce Error in Program Development; **IBM System Journal 15**; 1976

FBC 94a **Especificação do Domínio do Problema**; Projeto SEC (Sistemas Especialistas em Cardiologia); Fundação Bahiana de Cardiologia; julho 1994

FBC 94b **Plano do Projeto Sistemas Especialistas em Cardiologia**; Projeto SEC (Sistemas Especialistas em Cardiologia); Fundação Bahiana de Cardiologia; julho 1994

Fenton 91 Fenton, N.E.; **Software Metrics - A Rigorous Approach**; Chapman & Hall; 1991

Fernandes 94 Fernandes, A.K.; "Informatização de Banco de Sangue"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Finep/FBC **Projeto Desenvolvimento de Sistemas Especialistas em Cardiologia**; FINEP/ FBC; março 1994

Fiorini 94 Fiorini, S.T. e Leite, J.C.; "Qualidade na Definição de Requisitos"; **Workshop Qualidade de Software**; Curitiba, PR; outubro 1994

Fountoura 94 Fountoura, L.da e Alves, M.V.; "Computer Aided Educational Software for Teaching Biological Vision"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Franklin 88 Franklin, W.R., Gilbert, R.B. e Shroff, G.; "Debugging and Tracing Expert System"; **Proceedings of the Twenty-first Annual Hawaii International Conference on System Sciences**; 1988

Freedman 84 Freedman, D. P. and Weinberg, G. M.; "Reviews, Walkthroughs, and Inspections"; **IEEE Trans. Software Engineering**; janeiro 1984

Frewin 86 Frewin, G.D. e Hatton, B.J.; "Quality Management - procedures and practices"; **Software Engeneering Journal**; janeiro 1986

Furukawa 94 Furukawa, C. et al; "Computer Aided Surgery System for Stereotactic Neurosurgery"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Garcia 91 Garcia, O.N. e Chien, Y.; **Knowledge-Based Systems: Fundamentals and Tools**; IEEE Computer Society Press; 1991

Geissman 88 Geissman, J.R.e Schultz, R.D.; "Verification and Validation of Expert Systems"; **AI Expert**; fevereiro 1988

Giakoumakis 93 Giakoumakis, E. e Diamantidis, N.; "System for ECG Diagnosis Expert Systems"; **Computers in Cardiology**; IEEE Computer Society Press; Londres, Inglaterra; setembro 1993

Ginsberg 88 Ginsberg, A.; "Knowledge-Based Reduction: A New Approach to Checking Knowledge Bases for Inconsistency & Redundancy"; **Seventh National Conference on AI**; agosto 1988

Gottgtroy 90 Gottgtroy, M.P.B.; **O Processo de Aquisição do Conhecimento na Construção de Sistemas Especialistas**; Tese de Mestrado, COPPE/UFRJ; 1990

Gowen 93 Gowen, L.D. e Yap, M.Y.; "Traditional Software Development's Effects on Safety"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Green 87 Green, C.J.R. e Keyes, M.M.; "Verification and Validation of Expert Systems"; **Western Conference on Expert Systems**; 1987

Guida 93 Guida, G. e Mauri, G.; "Evaluating Performance and Quality of Knowledge-Based Systems: Foundation and Methodology"; **IEEE Transactions on Knowledge and Data Engineering;** vol. 5, no 2; abril 1993

Guida 94 Guida, G. e Tasso, C.; **Design and Develepment of Knowledge-Based Systems - From Life Cycle to Methodology**; Wiley Professional Computing, cap.9; 1994

Halstead 77 Halstead, M.H.; **Elements of Software Science**; Elsevier North-Holland; 1977

Harrison 94 Harrison, P.R. e Harrison, P.A.; "Validating an Embedded Sensor Control System"; **IEEE Expert**; junho 1994

Hayes-Roth 94 Hayes-Roth, F. e Jacobstein, N.; "The State of Knowledge-Based Systems"; **Communications of the ACM**, Knowledge Engineering Systems; março 1994

Herren 93 Herren, L.T. et al; "A Multi-Agent Knowledge-Based System for Planning and Scheduling Medical Tests"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Hickman 89 Hickman, F.R. et al; **Analysis for Knowledge-Based Systems - A Pratical Guide to the KADS Methodology**; Ellis Horwood; 1989

Hoare 78 Hoare, C.A.R.; "An axiomatic basis for computer programming"; **Programming Methodology**; Gries, D. (ed.); 1978

Hoefel 94 Hoefel, H.K. et al; "Sistema de Controle de Infecções Hospitalares Informatizado"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Hoppe 93 Hoppe, T. e Meseguer, P.; "VVT Terminology: A Proposal"; **IEEE Expert**; junho 1993

Hull 91 Hull, L.G.; Kay,P. "Expert Systems Development and Management"; **International Conference on Developping and Managing Expert Systems Programs**; Washington, USA; 1991

Ince 90 Ince, D.; "Software metrics: introduction"; **Information and Software Tecnology**; vol. 2, nº 1; maio 1990

Ip 93 Ip, H.H. et al; "Quality Function Deployment for a Medical Device"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

ISO 9000-3 ISO-9000-3; **Quality Management and Quality Assurance Standards - Part 3: Guidelines for the Application of ISO9001 to the Development, Supply and Maintenance of Software**; ISO; setembro 1990

ISO/CD 8402 ISO/CD 8402; **Quality Concepts and Terminology Part One: Generic Terms and Definition**; ISO; dezembro 1990

ISO/IEC 9126 ISO/IEC 9126; **Information Technology - Software Evaluation - Quality Characteristics and Guidelines for their use**; ISO; dezembro 1991

Jack 93 Jack, R.; "Applying ISO 9001 to Small scale Software Development Projects"; **IEEE Software**; 1993

Jafar 93 Jafar, M. e Bahill, A.T.; "Interactive Verification of Knowledge-Based Systems"; **IEEE Expert**; fevereiro 1993

James 93 James, T.L. Magee, M e Bruyere, H.J. ; "A Computer Imaging System for Noninvasively Determining Blood Pool Volume in the Chick Embryo Heart"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Jao 93 Jao, C.S. e Hier, D.B.; "Quality Function Deployment for a Medical Device"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Kaempf 94 Kaempf, L. et al; "CIDI-10/AUTO: Classificação Internacional de Doenças - Transtornos Mentais (10º visão Informatizada)"; **IV congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Kane 88 Kane, B. e Rucker, D.W.; "AI in Medicine"; **AI Expert**; 1988

Khuwaja 93 Khuwaja, R.A. et al; "Building the Domain-Expert for a Cardiovascular Physiology Tutor"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Kim 93 Kim, P.T.H.; "FDA and the Regulation on Medical Software"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Kiper 92 Kiper, J.D.; "Structural Testing of Rule-Based Expert Systems"; **ACM Transactions on Software Engineering and Methodology**; vol 1, No 2; 1992

Klempt 94 Klempt, A. e Infantosi, A.F.C.; "A Tutorial Microcomputer-Based System for Human Skull Anatomy"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Knight 93 Knight, J.C. e Myers, E.A.; "An Improved Inspection technique"; **Communications of the ACM**; novembro 1993

Kozinska 94 Kozinska, D. e Tarnecki, R.; "3D Imaging of Brain Lesions"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Krishnamurthy 87 Krishnamurthy, S.P. e Sztipanovits, J.; "Methodology for Testing and Validating Knowledge Bases"; **Proceedings of the Third Conference on Artificial Inteligence for Space Applications**; novembro 1987

Leão 93 Leão, B. de F.; Reategui, R.B. "A Hibrid Connectionist Expert System to Solve Classification Problems"; **Computers in Cardiology**; IEEE Computer Society Press; Londres, Inglaterra; setembro 1993

Leffingwell 93 Leffingwell, D.A. e Norman, B.; "Software Quality in Medical Devices - A Top-Down Approach"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Levenson 90 Levenson, N.G.; "Evaluation of Software Safety"; **Internacional Conference in Software Engeneering**; Nice; 1990

Linberg 93 Linberg, K.R.; "Defining the Role of Software Quality Assurance in a Medical Device Company"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Linger 79 Linger, R. et al; **Structured Programming**; Addison-Wisley; 1979

Luger 89 Luger, G. e Stubblefield, W.A.; **Artificial Inteligence and Design of Expert Systems**; Benjamim Cummings Publishing Company; 1989

Ma 93 Ma, H-N.N. et al; "An Intelligent Hypermedia System for Generating Progress Botes and Physician Reminders"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Mallory 93 Mallory, P. E.; "Building Quality into Medical Product Software Design"; **Biomedical Instrumentation & Technology**; maio/junho 1993

Marcot 87 Marcot, B.; "Testing Your Knowledge Base"; **AI Expert**; agosto 1987

Mazur 93 Mazur, G.; "Quality Function Deployment for a Medical Device"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

McCall 79 McCall, J., Cooper, J. D. e Fisher, M. J.; "An Introduction to Software Quality Metrics" in **Software Quality Management**; Petrocelli; 1979

Menezes 94 Menezes, H.S. et al; "MEDCIR - Sistema de Apoio à Decisão para Consultório"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Mengshoel 93 Mengshoel, O.J. e Delab, S.; "Knowledge Validation: Principles and Practice"; **IEEE Expert**; junho 1993

Merlevede 91 Merlevede, P. e Vanthienen, J.; "A Structured Approach to Formalization and Validation of Knowledge"; **IEEE Expert**; 1991

Meseguer 92 Meseguer, P. e Plaza, E.; "Validating of KBS: The VALID project""; **Enhancing the Knowledge Process**; North-Holland; 1992

Meseguer 93 Meseguer, P.; "Conventional Software and Expert Systems: Some Comparative Aspects Regarding Validation"; **Knowledge Oriented Software Design**; North-Holland; 1993

Micheli-Tzanakou 93 Micheli-Tzanakou, E. et al; "Myocardial Infarction: Diagnosis and Vital Status Prediction Using Neural Networks"; **Computers in Cardiology**; IEEE Computer Society Press; Londres, Inglaterra; setembro 1993

Miller 93 Miller, S.E., Tucker, G.T. e Verducci Jr., A.J.; "Quality Standards: The Role of Software process Assessments"; **IEEE Software**; 1993

Möller 93 Möller, K.H. e Paulish, D.J.; **Software Metrics - A Practioner's Guide to Improved Product Development**; IEEE Computer Society Press, Chapmam & Hall; 1993

Morell 88 Morell, L.J.; "Use of Metaknowledge in the Verification of Knowledge-based Systems"; **Proceedings of the IEA-AIE**; junho 1988

Morozoff 94 Morozoff,P.E.; "Specifying Software Validation and Verification for a Biomedical Application"; **Biomedical Instrumentation & Technology**; maio/junho 1994

Moura 94 Moura Jr.,J.M.A. et al; "Suprarenal and Glaucoma Computer Assisted Instruction"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Musa 87 Musa, J.D., Iannino, A. e Okumoto,K.; "Engineering and Managing Software with Reliability Measures"; McGraw-Hill; 1987

Nadal Nadal, J. e Bossan,M.C.; "Classification of Cardiac Arrhytmias Based on Principal Caomponent Analysis and Feedforward Neural Networks"; **Computers in Cardiology**; IEEE Computer Society Press; Londres, Inglaterra; setembro 1993

Naser 88 Naser, J.A.; "Nuclear Power Plant Expert System Verification and Validation"; **Proceedings of the Workshop on Validation and Verification of Expert System**; agosto 1988

Ng 93 Ng, J.M. et al; "A Multimedia Conferencing System for Co-Operative Medical Diagnosis"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Nguyen 87 Nguyen, T.A., Perkins, W.A., Laffey, T.J. e Pecora, D.; "Knowledge Base Verification"; **AI Magazine**; 1987

Nieberl 93 Nieberl, J. et al; "A Hibrid Connectionist Expert System to Solve Classification Problems"; **Computers in Cardiology**; IEEE Computer Society Press; Londres, Inglaterra; setembro 1993

O'Keefe 87 O'Keefe, R.. Balci, O. e Smith, E.; "Validating Expert Systems Performance"; **IEEE Expert**; 1987

O'Leary 87 O'Leary, D.E.; "Validation of Expert Systems - with Application to Auditing and Accounting Expert Systems"; **Decision Sciences**; 1987

O'Leary 90 O'Leary, T.J., Goul, M.G., Moffitt, K.E. e Radwan, A.E.; "Validating Expert Systems"; **IEEE Expert**; junho 1990

Oliveira 94a Oliveira, K.M., Werneck, V.M.B. e Rocha, A.R.C.da; "Avaliação da Qualidade de Sistemas Especialistas"; **Workshop Qualidade de Software**; Curitiba, PR; 1994

Oliveira 94b Oliveira, K.M., Werneck, V.M. e Rocha, A.R.; "Definição dos Requisitos de Qualidade de um Sistema Especialista em Cardiologia"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Palermo 89 Palermo, S. e Rocha, A.R.C.da; "An experience on Evaluating Software Quality for High energy Physics"; **Computer Physics Communications**; 1989

Palombo 94 Palombo, C.R. et al; "DIAGFACE: Um banco de conhecimento e sistema especialista para o diagnóstico de patologias orofaciais"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Passos 91 Passos, M.C.J.F.; **Uma Ferramenta para Construção e Avaliação de Gráficos de Estrutura**; Tese de Mestradom COPPE/UFRJ; maio 1991

Passos 92 Passos, M.C.J.F. e Rocha, A.R.C.da; "Uma ferramenta para Avaliação de Gráficos de Estrutura"; **XII Congresso de Metodologias en Ingenieria de Sistemas**; Santiago, Chile, julho 1992

Passos 94 Passos, M.C.J.F. e Rocha, A.R.C.da; "Um Assistente Baseado em Conhecimento para Apoio ao Planejamento da Qualidade de Projetos de Desenvolvimento de Software"; **Workshop Qualidade de Software**; Curitiba, PR; outubro 1994

Patterson 90 Patterson, D. W.; "Introduction to Artificial Intelligence and Expert Systems"; **IEEE Expert**; 1993

Pecoits 94 Pecoits,A.P. et al ; "Speech Recognition using Zero Detection Technique"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Polat 93 Polat, F. e Guvenir, H.A.; "UVT: A Unification-Based Tool for Knowledge Base Verification"; **IEEE Expert**; junho 1993

Pressman 92 Pressman, R.; **Software Engineering - A Practioner's Approach**; Third Edition; McGraw Hill International Edition; 1992

Prosdocimo 94 Prosdocimo, M.Z., Asinelli, P.A. e Nohama, P.; "A Voice-pattern Recognition to Command Neuro-muscular Stimulator"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Pryor 94 Pryor, T.A.; "HELP: A Patient-Centered Hospital Information System"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Rabelo 94a Rabelo, A. Jr. et al; "Uma Experiência na Definição, Uso e Avaliação do Processo de Desenvolvimento de Sistemas Especialistas em Cardiologia"; **Workshop Qualidade de Software**; Curitiba, PR; outubro 1994

Rabelo 94b Rabelo, A. Jr. et al; **Avaliação do 1º Estágio do Projeto Sistemas Especialistas em Cardiologia**; Publicações Técnicas, Núcleo de Pesquisa e Desenvolvimento de Software Médico - FBC; 1994

Rabelo 95 Rabelo, A.Jr. et al; "An Expert System for Diagnosis of Acute Myocardial Infarction", **ACM Symposium on Applied Computing**; Nashville, Estados Unidos; fevereiro 1995

Robert 94 Robert, L., Maingourd, Y e Lerallut, JF.; "Development and Optmisation of a Workstation for the Evaluation of Cardiac Function in Children"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Rocha 83 Rocha,A.R.C.da; **Um Modelo para Avaliação da Qualidade de Especificações**, Tese de Doutorado, PUC/RJ, junho 1983

Rocha 87 Rocha,A.R.C.da; **Análise e Projeto Estruturado de Sistemas**; Ed. Campus; 1987

Rocha 94 Rocha, A.R.C.da et al; "Uma Experiência na Definição do Processo de Desenvolvimento e Avaliação de Software segundo as Normas ISO"; **Workshop Qualidade de Software**, Curitiba, PR; outubro 1994

Rook 86 Rook, P.; "Controlling Software Projects"; **Software Engineering Journal**; vol. 1., nº 1; January 1986

Rushby 88 Rushby, J.; "Validation and Testing of Knowledge-Based Systems - How bad can it get?"; **AAAI-88 Workshop on Validation and Verification of Expert Systems**; dezembro 1988

Sanders 94 Sanders, J. e Curran, E.; **Software Quality - A Framework for Success in Software Development and Support**; Addison-Wesley Publishing Company; 1994

Saranummi 91 Saranummi, N. et al; "Knowledge-based Systems in Medicine- a Nordic research and development programme"; **Computer Methods and Programs in Biomedicine**; Elsevier Science Publishers; 1991

Schreiber 92 Schreiber,G (ed) Knowledge Acquisition: **The KADS Approach for Knowledge Engineering**; Academic Press; 1992

Schultz 88 Schultz, R.D. e Geissman, J.R.; "Bridging the Gap Between Static and Dynamic Verification"; **Validation and Testing Knowledge-Based Systems Workshop**; agosto 1988

Scott 91 Scott, A. C., Clayton, J. E. e Gibson, E. L.; **A Pratical Guide to Knowledge Acquisition**; Massachusetts; Addison Wesley; 1991

Sefcik 94 Sefcik, J.G.; "Critical Success Factors Implementing Software Quality Plans"; **Software Engineering Notes**; vol 19, nº 1; 1994

Selby 87 Selby, R.W., Basili, V.R. e Baker, F.T.; "Cleanroom software Development: an Empirical Evaluation"; **IEEE Trans. Software Engineering**; setembro 1987

Shepperd 92 Shepperd, M.; "Products, process and metrics"; **Information and Software Tecnology**; vol 34, nº 10; outubro 1992

Shih-Min 94 Shih-Min, L. et al; "Realce e segmentação de imagens médicas usando o sistema Khoros"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Silva 94a Silva, A.M.M., D'Ornellas, M..C. e Tamiosso, F.S.; "Realce e segmentação de imagens médicas usando o sistema Khoros"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Silva 94b Silva, I.P.daR.; "Sistema de Prescrição Médica"; **IV Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994

Sorace 93 Sorace, J.M. e Moore, G.W.; "Rapid Validation of a Medical Expert System Using Precedence Logic"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Spitzer 93 Spitzer, A.R. e Bearden, F.; "DANIEL: A Dinamic Acquisition & Numerical Information Environment Language and RIVKA: Robust Intelligent Vector Krunching Application"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Stachowitz 87 Stachowitz, R.A. e Combs, J.B.; "Validating of Expert Systems"; **The Proceedings of the 20th Hawaii Internacional Conference on Systems Science**; 1987

Starita 94 Starita, A. et al; "A Tutorial System to support diagnosis of neurogenic pathologies at the inferior limbs"; **World Congress on Medical Physics and Biomedical Engineering**; Rio de Janeiro, RJ; agosto 1994

Sukthankar 93 Suthankar, S. et al; "Graphical User Interfacing with Domain Visualization for an Intelligent Medical Tutoring System"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Suwa 82 Suwa, M., Scott, A,C. e Shortliffe, E.H.; "An Approach to Verifying Completeness and Consistency in Rule-Based Expert Systems"; **AI Magazine**; 1982

Taddei 93 Taddei, A. et al; "A Hibrid Connectionist Expert System to Solve Classification Problems"; **Computers in Cardiology**; IEEE Computer Society Press; Londres, Inglaterra; setembro 1993

Tisdell 93 Tisdell, M.L. e Jerabek, C.F.; "Computerized Cardopulmonary Perfusion"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Turban 91 Turban, E.; "Managing Knowledge Acquisition from Multiple Experts"; **International Conference on Developing and Managing Expert Systems Programs**; Washington, Estados Unidos; outubro 1991

Voas 93 Voas, J., Miller, K. e Payne, J.; "A Software Analysis Technique for Quantifying Reliability in High-Risk Medical Devices"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; junho1993

Waterman 86 Waterman, D.A.; **A Guide to Expert Systems**; Addison-Wesley Publishing Company; 1986

Weber 94a Weber, K. et al; **Qualidade e Produtividade em Software: Termo de Referência do Subprograma Setorial da Qualidade e Produtividade em Software, do Programa Brasileiro de Qualidade e Produtividade**; QA&T; 1994

Weber 94b Weber, K.C., Frizanco, O. e Graf, R.; "Dois Modelos para Gestão da Qualidade de Processos de Software"; **Workshop Qualidade de Software**; Curitiba, PR; outubro 1994

Werneck 94 Werneck,V.M., Oliveira, K.M.de; Rocha, A.R.C.da; **Processo de Desenvolvimento do Projeto Sistemas Especialistas em Cardiologia**; Publicações Técnicas, COPPE/UFRJ; agosto 1994

Werneck 95 Werneck, V.M.B.; **Ambiente de Desenvolvimento de Sistemas Baseados em Conhecimento**; Tese de Doutorado; COPPE/UFRJ; março 1995

Wielenga 88 Wielenga, B.J. et al; "Knowledge Acquisition for Expert Systems"; **Proceedings of ACAI**; 1988

Yourdon 89 Yourdon, E.; **Structured Walkthrough**; 4th edition; Yourdon Press (Prentice Hall); 1989

Zemlin 93 Zemlin, B. e Eagles, S.; "Establishing a Software Engeneering Process Group in a Medica Device Company"; **Computer-Based Medical Systems**, Proceedings of the Sixth Annual IEEE Symposium; IEEE Computer Society Press; Michigan, Estados Unidos; junho 1993

Zhang 94 Zhang, D. e Nguyen, D.; "PREPARE: A Tool for Knowledge Base Verification"; **IEEE Transactions on Knowledge and Data Engineering**, vol. 6, no. 6; dezembro 1994

Anexo I

Formulários para Avaliação da Modelagem

Critérios para Validação da Modelagem
Avaliação de Especialistas

Nome do Avaliador: _____

Aspectos relacionados à forma do documento

1) De forma geral, o documento está claro. (Marque uma resposta)

- Concordo plenamente ⊂⊃
- Concordo ⊂⊃
- Discordo ⊂⊃
- Discordo em parte ⊂⊃
- Discordo totalmente ⊂⊃
- Não tenho condições de avaliar ⊂⊃

2) Todos os termos de entendimento não óbvio estão definidos no glossário?

- Sim ⊂⊃
- Não ⊂⊃

Indique quais não estão:

_____
_____
_____
_____
_____

3) A terminologia, usada no documento, está sendo utilizada de forma uniforme?

- Sim ⊂⊃
- Não ⊂⊃

Em caso de resposta *Não,* indique onde:

_____
_____
_____
_____
_____

Aspectos relacionados ao conteúdo do documento

4) A modelagem do sistema *(seção 4)* inclui, apenas, os aspectos necessários?

Sim ⊂⊃

Não ⊂⊃

Identifique os aspectos não necessários:

_____

_____

_____

_____

_____

5) A modelagem do sistema *(seção 4)* apresenta aspectos redundantes?

Sim ⊂⊃

Não ⊂⊃

Identifique onde há aspectos redundantes:

_____

_____

_____

_____

_____

6) A modelagem do sistema *(seção 4)* apresenta aspectos contraditórios?

Sim ⊂⊃

Não ⊂⊃

Identifique os aspectos contraditórios:

_____

_____

_____

_____

_____

7) A camada de domínio do problema *(seção 4.1)* representa corretamente o que foi especificado?

Sim ⊂⊃

Não ⊂⊃

Indique o que está errado:

_____

_____

_____

_____

_____

8) A camada de inferência *(seção 4.2)* representa corretamente o que foi especificado?

Sim ⊂⊃

Não ⊂⊃

Indique o que está errado:

_____

_____

_____

_____

_____

9) A camada de tarefas *(seção 4.2)* representa corretamente o que foi especificado?

Sim ⊂⊃

Não ⊂⊃

Indique o que está errado:

_____

_____

_____

_____

_____

10) A camada de estratégia *(seção 4.3)* representa corretamente o que foi especificado?

Sim ⊂⊃

Não ⊂⊃

Indique o que está errado:

_____

_____

_____

_____

_____

Outros problemas encontrados no documento:

_____

_____

_____

_____

_____

_____

_____

_____

Critérios para Validação da Modelagem
Avaliação de Engenheiros de Software

Nome do Avaliador: _____

Aspectos relacionados à forma do documento

1) De forma geral, o documento está claro. (Marque uma resposta)

- Concordo plenamente ⊂⊃
- Concordo ⊂⊃
- Discordo ⊂⊃
- Discordo em parte ⊂⊃
- Discordo plenamente ⊂⊃
- Não tenho condições de avaliar ⊂⊃

2) Todos os termos de entendimento não óbvio estão definidos no glossário?

- Sim ⊂⊃
- Não ⊂⊃

Indique quais não estão:

_____
_____
_____
_____
_____

3) De forma geral, o documento está conciso?

- Sim ⊂⊃
- Não ⊂⊃

4) A terminologia, usada no documento, está sendo utilizada de forma uniforme?

Sim ⊂⊃

Não ⊂⊃

Em caso de resposta *Não,* indique onde:

_____

_____

_____

_____

_____

Aspectos relacionados ao conteúdo do documento

5) O conhecimento está representado *(seção 4.1)* corretamente utilizando redes semânticas?

Sim ⊂⊃

Não ⊂⊃

Indique o que está errado:

_____

_____

_____

_____

_____

6) A camada de inferência *(seção 4.2)* está representada, corretamente, de acordo com o método KADS?

Sim ⊂⊃

Não ⊂⊃

Indique o que está errado:

_____

_____

_____

_____

_____

7) A camada de tarefas *(seção 4.3)* está representada, corretamente, de acordo com o método KADS?

Sim ⊂⊃

Não ⊂⊃

Indique o que está errado:

_____

_____

_____

_____

_____

8) A camada de estratégia *(seção 4.3)* está representada, corretamente, de acordo com o método KADS?

Sim ⊂⊃
Não ⊂⊃

Indique o que está errado:

_____
_____
_____
_____
_____

9) De forma geral, as quatro camadas *(seção 4)* estão relacionadas corretamente. (Marque uma resposta)

Concordo plenamente ⊂⊃
Concordo ⊂⊃
Discordo ⊂⊃
Discordo em parte ⊂⊃
Discordo plenamente ⊂⊃
Não tenho condições de avaliar ⊂⊃

10) As seções do documento, estabelecidas no Plano de Documentação para esta fase, estão com um mesmo nível de detalhe?

Sim ⊂⊃
Não ⊂⊃

Identifique onde não satisfaz:

_____
_____
_____
_____
_____

Outros problemas encontrados no documento:

_____
_____
_____
_____
_____
_____
_____
_____

Anexo II
Plano de Controle da Qualidade

Projeto SEC
Plano de Controle de Qualidade

1. Controle da Qualidade ao longo do desenvolvimento

A avaliação da qualidade do produto se fará ao longo de todo o processo de desenvolvimento. Durante as diversas etapas do projeto existirão dois tipos de avaliações:

- Avaliações Intermediárias
- Avaliação Final do Produto da Etapa

Estas avaliações serão realizadas através da técnica de ***inspeções por fases*** [Knight e Myers, 1993], conforme descrito no documento "Processo de Desenvolvimento do SEC".

Avaliações Intermediárias podem ser realizadas através de inspeções com inspetores individuais ou de reuniões de inspeção. Os participantes, em cada avaliação, serão definidos caso a caso.

Uma *Avaliação Final do Produto da Etapa*, deve, sempre, ser realizada ao se dar por concluidas as tarefas de uma etapa e antes de se passar à próxima etapa. Esta avaliação tem como objetivo obter-se a aprovação do usuário para o produto da etapa. Esta aprovação é formalizada pelo aceite do coordenador do projeto na FBC.

A avaliação final do produto é feita, obrigatoriamente, através de uma reunião de inspeção da qual devem participar:

- o coordenador do projeto na FBC ou um representante por ele designado;
- um engenheiro de software da COPPE
- pelo menos dois especialistas (nas primeiras versões do sistema podem ser apenas especialistas que participaram no desenvolvimento. Em versões posteriores pelo menos um especialista não deve ter participado do desenvolvimento)
- o coordenador técnico do projeto
- dois representantes dos desenvolvedores
- pelo menos dois representantes dos potenciais usuários (em versões finais do SEC)

Além destes, podem ser solicitados a participar a critério do coordenador do projeto:

- representantes da COPPE, que venham participando do projeto
- outros avaliadores que não tenham participado do desenvolvimento

1.1. Controle da Qualidade na Etapa de Análise do Domínio do Problema

1. Validação do Conhecimento Geral

Responsável: Coordenador técnico do projeto

Sistemática:

- **inspeção com inspetores individuais** a ser realizada pelos engenheiros de software segundo os Critérios para Avaliação do Conhecimento Geral (avaliação de engenheiros de software)
- **reunião de inspeção** com participação do coordenador técnico, especialistas, representantes dos desenvolvedores segundo os Critérios para Avaliação do Conhecimento Geral (avaliação de especialistas e avaliação do coordenador técnico)

2. Avaliação da Especificação do Domínio do Problema (Avaliação de produto final de etapa)

Responsável: Coordenador do projeto

Sistemática:

- **inspeção com inspetores individuais** a ser realizada pelos engenheiros de software segundo os Critérios para Avaliação da Especificação do Domínio do Problema (avaliação de engenheiros de software)
- **reunião de inspeção** com participação do coordenador do projeto, do coordenador técnico, especialistas, representantes dos desenvolvedores e outros avaliadores segundo os Critérios para Avaliação da Especificação do Domínio do Problema (avaliação do coordenador do projeto, avaliação de especialistas, avaliação do coordenador técnico e outros avaliadores)

1.2 Controle da Qualidade na Etapa de Planejamento do Projeto

1. Avaliação do Plano do Projeto (Avaliação de produto final de etapa)

Responsável: Coordenador do projeto

Sistemática:

- **reunião de inspeção** com participação do coordenador do projeto, engenheiro de software da COPPE, coordenador técnico e representantes dos desenvolvedores segundo os Critérios para Avaliação do Plano do Projeto (avaliação do coordenador do projeto e avaliação do coordenador técnico)

1.3 Controle da Qualidade nas Etapas de Desenvolvimento das Versões

Durante o desenvolvimento de cada estágio são realizadas as seguintes avaliações:

1. Validação dos Requisitos
Responsável: Coordenador técnico do projeto
Sistemática:

- **inspeção com inspetores individuais** a ser realizada pelos engenheiros de software segundo os Critérios para Validação dos Requisitos (avaliação de engenheiros de software)
- **reunião de inspeção** com participação do coordenador técnico, especialistas, representantes dos desenvolvedores segundo os Critérios para Validação dos Requisitos (avaliação de especialistas e avaliação do coordenador técnico)

2. Validação da Modelagem
Responsável: Coordenador técnico do projeto
Sistemática:

- **inspeção com inspetores individuais** a ser realizada pelos consultores segundo os Critérios para Validação da Modelagem (avaliação de consultores)
- **reunião de inspeção** com participação do coordenador técnico, especialistas, representantes dos desenvolvedores e outros avaliadores segundo os Critérios para Validação da Modelagem (avaliação de especialistas, avaliação do coordenador técnico e outros avaliadores)

3. Avaliação da Especificação de Requisitos (Avaliação de produto final de etapa)
Responsável: Coordenador do projeto
Sistemática:

- **inspeção com inspetores individuais** a ser realizada pelos engenheiros de software segundo os Critérios para Avaliação da Especificação de Requisitos (avaliação de engenheiros de software)
- **reunião de inspeção** com participação do coordenador do projeto, engenheiro de software da COPPE, do coordenador técnico, especialistas, representantes dos desenvolvedores e outros avaliadores segundo os Critérios para Avaliação da Especificação de Requisitos (avaliação do coordenador do projeto, avaliação de especialistas, avaliação do coordenador técnico e outros avaliadores)

4. Avaliação da Especificação de Projeto (Avaliação de produto final de etapa)
Responsável: Coordenador do projeto
Sistemática:

- **inspeção com inspetores individuais** a ser realizada pelos engenheiros de software segundo os Critérios para Avaliação da Especificação de Projeto (avaliação de engenheiros de software)
- **reunião de inspeção** com participação do coordenador do projeto, engenheiro de software da COPPE, do coordenador técnico, representantes dos desenvolvedores e outros avaliadores segundo os Critérios para Avaliação da Especificação de Projeto (avaliação do coordenador do projeto, avaliação do coordenador técnico e outros avaliadores)

2. Avaliação do Produto Final

O desenvolvimento do SEC é realizado, incrementalmente, em diferentes estágios de desenvolvimento. Por isso o sistema, progressivamente, incorporará o conhecimento e os requisitos desejados. Assim os critérios para avaliação da qualidade do produto final deverão ter intensidades diferentes para cada estágio do desenvolvimento, sendo especificados ao se planejar a qualidade de cada versão em desenvolvimento.

A aceitação do produto final de um estágio (uma versão do sistema) implica na aceitação do programa e de sua documentação. É responsabilidade do coordenador do projeto.

É realizada através da seguinte sistemática:

- **testar a versão N** como um todo, com os casos de teste elaborados pelos especialistas que participaram do desenvolvimento;

- **testar a versão N com o usuário,** durante esta atividade o sistema é submetido a testes na presença de um grupo de potenciais usuários para verificar se o sistema atingiu os requisitos estabelecidos na ERS (este teste só será realizado nas versões finais do sistema);

- **testar a versão N para aceitação na presença do coordenador do projeto**.

- **reunião de inspeção** com a participação do coordenador do projeto, do engenheiro de software da COPPE, do coordenador técnico, especialistas, representantes dos

desenvolvedores e outros avaliadores segundo os Critérios para Avaliação do Produto da Versão (avaliação do coordenador do projeto, avaliação de especialistas, avaliação do coordenador técnico e outros avaliadores);

A partir dos resultados obtidos nesta avaliação o coordenador do projeto na FBC dá o aceite ao sistema

3. Avaliação do Processo de Desenvolvimento

Esta avaliação é de responsabilidade do coordenador do projeto e do engenheiro de software da COPPE.

É realizada através de uma reunião de inspeção, com participação do coordenador do projeto, coordenador técnico, especialistas, desenvolvedores e de pelo menos um engenheiro de software da COPPE, segundo os critérios para avaliação do processo (avaliação do coordenador do projeto, coordenador técnico, especialistas, desenvolvedores e engenheiros de software).

Anexo III
Formulários para Avaliação da 1ª Versão

Avaliação da 1ª. versão do SEC

Avaliação do Coordenador do Projeto

Foram identificados, como importantes para a 1ª. versão do SEC, os seguintes requisitos de qualidade. Neste momento em que concluimos a 1ª. versão devemos avaliar o grau com que cada um deles foi atingido.

Após examinar o produto, assinale na escala o valor que lhe parece melhor representar o grau com que cada requisito foi atingido.

NOME DO AVALIADOR: _____

Facilidade de aprendizado	refere-se à facilidade para entendimento e utilização do sistema por usuários finais iniciantes |___|___|___|___|___| 0 1 2 3 4
Amenidade de Uso	característica do sistema apresentar os resultados de forma clara, ser capaz de disponibilizar auxílio de forma rápida e apresentar estabilidade no uso |___|___|___|___|___| 0 1 2 3 4
Interatividade	característica do sistema manter um diálogo conciso e claro com o usuário final |___|___|___|___|___| 0 1 2 3 4
Oportunidade	característica do sistema produzir resultados, em situações típicas, no tempo adequado |___|___|___|___|___| 0 1 2 3 4

Eficiência de execução	característica do sistema ser executado no menor tempo possível |___|___|___|___|___| 0 1 2 3 4
Grau de explicação	característica do sistema prover uma justificativa da solução gerada |___|___|___|___|___| 0 1 2 3 4
Adequabilidade	característica de ser adequada a construção do sistema, considerando-se a natureza, a complexidade e o escopo do problema |___|___|___|___|___| 0 1 2 3 4
Confiabilidade da implementação	característica do sistema proporcionar soluções únicas e suficientemente detalhadas de modo a satisfazer a utilização pretendida pelo usuário |___|___|___|___|___| 0 1 2 3 4
Especificidade	característica da especificação do sistema abordar, apenas os requisitos necessários e de sua implementação gerar, apenas, as soluções especificadas |___|___|___|___|___| 0 1 2 3 4
Não redundância	característica da especificação do sistema não conter conhecimentos nem gerar soluções redundantes |___|___|___|___|___| 0 1 2 3 4
Consistência	característica do sistema não apresentar conhecimentos contraditórios, nem soluções contraditórias em sua implementação |___|___|___|___|___| 0 1 2 3 4
Uniformidade de terminologia	característica do sistema estar documentado com uniformidade de notação e padronização de termos técnicos |___|___|___|___|___| 0 1 2 3 4

Disponibilidade da documentação	característica do sistema ter sua documentação atualizada e pronta para uso quando necessário |___|___|___|___|___| 0 1 2 3 4

Avaliação da 1ª. versão do SEC

Avaliação de Especialista

Foram identificados, como importantes para a 1ª. versão do SEC, os seguintes requisitos de qualidade. Neste momento em que concluimos a 1ª. versão devemos avaliar o grau com que cada um deles foi atingido.

Após examinar o produto, assinale na escala o valor que lhe parece melhor representar o grau com que cada requisito foi atingido.

NOME DO AVALIADOR: _____

Facilidade de aprendizado	refere-se à facilidade para entendimento e utilização do sistema por usuários finais iniciantes |___|___|___|___|___| 0 1 2 3 4
Amenidade de Uso	característica do sistema apresentar os resultados de forma clara, ser capaz de disponibilizar auxílio de forma rápida e apresentar estabilidade no uso |___|___|___|___|___| 0 1 2 3 4
Interatividade	característica do sistema manter um diálogo conciso e claro com o usuário final |___|___|___|___|___| 0 1 2 3 4
Oportunidade	característica do sistema produzir resultados, em situações típicas, no tempo adequado |___|___|___|___|___| 0 1 2 3 4

Eficiência de execução	característica do sistema ser executado no menor tempo possível |___|___|___|___|___| 0 1 2 3 4
Grau de explicação	característica do sistema prover uma justificativa da solução gerada |___|___|___|___|___| 0 1 2 3 4
Adequabilidade	característica de ser adequada a construção do sistema, considerando-se a natureza, a complexidade e o escopo do problema |___|___|___|___|___| 0 1 2 3 4
Confiabilidade da implementação	característica do sistema proporcionar soluções únicas e suficientemente detalhadas de modo a satisfazer a utilização pretendida pelo usuário |___|___|___|___|___| 0 1 2 3 4
Especificidade	característica da especificação do sistema abordar, apenas os requisitos necessários e de sua implementação gerar, apenas, as soluções especificadas |___|___|___|___|___| 0 1 2 3 4
Não redundância	característica da especificação do sistema não conter conhecimentos nem gerar soluções redundantes |___|___|___|___|___| 0 1 2 3 4
Consistência	característica do sistema não apresentar conhecimentos contraditórios, nem soluções contraditórias em sua implementação |___|___|___|___|___| 0 1 2 3 4
Uniformidade de terminologia	característica do sistema estar documentado com uniformidade de notação e padronização de termos técnicos |___|___|___|___|___| 0 1 2 3 4

Avaliação da 1ª. versão do SEC

Outros Avaliadores

Foram identificados, como importantes para a 1ª. versão do SEC, os seguintes requisitos de qualidade. Neste momento em que concluimos a 1ª. versão devemos avaliar o grau com que cada um deles foi atingido.

Após examinar o produto, assinale na escala o valor que lhe parece melhor representar o grau com que cada requisito foi atingido.

NOME DO AVALIADOR: _____

Facilidade de aprendizado	refere-se à facilidade para entendimento e utilização do sistema por usuários finais iniciantes |___|___|___|___|___| 0 1 2 3 4
Amenidade de Uso	característica do sistema apresentar os resultados de forma clara, ser capaz de disponibilizar auxílio de forma rápida e apresentar estabilidade no uso |___|___|___|___|___| 0 1 2 3 4
Interatividade	característica do sistema manter um diálogo conciso e claro com o usuário final |___|___|___|___|___| 0 1 2 3 4
Oportunidade	característica do sistema produzir resultados, em situações típicas, no tempo adequado |___|___|___|___|___| 0 1 2 3 4
Eficiência de execução	característica do sistema ser executado no menor tempo possível |___|___|___|___|___| 0 1 2 3 4

Grau de explicação	característica do sistema prover uma justificativa da solução gerada |___|___|___|___|___| 0 1 2 3 4
Adequabilidade	característica de ser adequada a construção do sistema, considerando-se a natureza, a complexidade e o escopo do problema |___|___|___|___|___| 0 1 2 3 4
Confiabilidade da implementação	característica do sistema proporcionar soluções únicas e suficientemente detalhadas de modo a satisfazer a utilização pretendida pelo usuário |___|___|___|___|___| 0 1 2 3 4
Especificidade	característica da especificação do sistema abordar, apenas os requisitos necessários e de sua implementação gerar, apenas, as soluções especificadas |___|___|___|___|___| 0 1 2 3 4
Não redundância	característica da especificação do sistema não conter conhecimentos nem gerar soluções redundantes |___|___|___|___|___| 0 1 2 3 4
Consistência	característica do sistema não apresentar conhecimentos contraditórios, nem soluções contraditórias em sua implementação |___|___|___|___|___| 0 1 2 3 4
Clareza	característica do sistema estar especificado, modelado e implementado da forma mais clara possível, isento de práticas que o tornem complexo e de difícil entendimento |___|___|___|___|___| 0 1 2 3 4
Disponibilidade da documentação	característica do sistema ter sua documentação atualizada e pronta para uso quando necessário |___|___|___|___|___| 0 1 2 3 4

Avaliação 1a. versão do SEC

Avaliação dos Desenvolvedores

Foram identificados, como importantes para a 1a. versão do SEC, os seguintes requisitos de qualidade. Neste momento em que concluimos a 1a. versão devemos avaliar o grau com que cada um deles foi atingido.

Após examinar o produto, assinale na escala o valor que lhe parece melhor representar o grau com que cada requisito foi atingido.

NOME DO AVALIADOR: _____

Adequação da metodologia	característica do sistema usar procedimentos e métodos adequados para a representação do conhecimento |___|___|___|___|___| 0 1 2 3 4
Adequação do ambiente de programação	característica do sistema ter sido implementado utilizando um ambiente de programação adequado às suas necessidades e que seja capaz de suportar futuras evoluções no sistema |___|___|___|___|___| 0 1 2 3 4
Codificabilidade	característica do sistema utilizar técnicas de representação do conhecimento, para codificação e atualização da base de conhecimentos, poderosas e flexíveis |___|___|___|___|___| 0 1 2 3 4
Clareza	característica do sistema estar especificado, modelado e implementado da forma mais clara possível, isento de práticas que o tornem complexo e de difícil entendimento |___|___|___|___|___| 0 1 2 3 4

Concisão	característica do sistema ter sido especificado e modelado de forma concisa e implementado com a quantidade mínima de comandos |___|___|___|___|___| 0 1 2 3 4
Modularidade	característica do sistema ser projetado e implementado através de uma estrutura de módulos independentes e particionados logicamente |___|___|___|___|___| 0 1 2 3 4
Correção da representação	característica do sistema estar correto do ponto de vista do uso da linguagem de especificação adotada no que se refere a notação, semântica, sintaxe e formato de documentação |___|___|___|___|___| 0 1 2 3 4
Uniformidade de terminologia	característica do sistema estar documentado com uniformidade de notação e padronização de termos técnicos |___|___|___|___|___| 0 1 2 3 4
Uniformidade no grau de abstração	característica da documentação do sistema possuir um nível uniforme de detalhe, considerando-se um determinado estágio do desenvolvimento |___|___|___|___|___| 0 1 2 3 4
Completude da documentação	característica da documentação estar completa de acordo com os roteiros estabelecidos no plano de documentação do projeto |___|___|___|___|___| 0 1 2 3 4
Disponibilidade da documentação	característica do sistema ter sua documentação atualizada e pronta para uso quando necessário |___|___|___|___|___| 0 1 2 3 4
Estrutura	característica do sistema possuir um padrão definido de composição de suas partes, formando uma organização hierárquica de regras e/ou conhecimento |___|___|___|___|___| 0 1 2 3 4

Avaliação da 1ª. versão do SEC

Avaliação de Engenheiros de Software

Foram identificados, como importantes para a 1ª. versão do SEC, os seguintes requisitos de qualidade. Neste momento em que concluimos a 1ª. versão devemos avaliar o grau com que cada um deles foi atingido.

Após examinar o produto, assinale na escala o valor que lhe parece melhor representar o grau com que cada requisito foi atingido.

NOME DO AVALIADOR: _____

Facilidade de aprendizado	refere-se à facilidade para entendimento e utilização do sistema por usuários finais iniciantes |___|___|___|___|___| 0 1 2 3 4
Amenidade de Uso	característica do sistema apresentar os resultados de forma clara, ser capaz de disponibilizar auxílio de forma rápida e apresentar estabilidade no uso |___|___|___|___|___| 0 1 2 3 4
Interatividade	característica do sistema manter um diálogo conciso e claro com o usuário final |___|___|___|___|___| 0 1 2 3 4
Oportunidade	característica do sistema produzir resultados, em situações típicas, no tempo adequado |___|___|___|___|___| 0 1 2 3 4
Eficiência de execução	característica do sistema ser executado no menor tempo possível |___|___|___|___|___| 0 1 2 3 4

Grau de explicação	característica do sistema prover uma justificativa da solução gerada |___|___|___|___|___| 0 1 2 3 4
Adequação da metodologia	característica do sistema usar procedimentos e métodos adequados para a representação do conhecimento |___|___|___|___|___| 0 1 2 3 4
Adequação do ambiente de programação	característica do sistema ter sido implementado utilizando um ambiente de programação adequado às suas necessidades e que seja capaz de suportar futuras evoluções no sistema |___|___|___|___|___| 0 1 2 3 4
Codificabilidade	característica do sistema utilizar técnicas de representação do conhecimento, para codificação e atualização da base de conhecimentos, poderosas e flexíveis |___|___|___|___|___| 0 1 2 3 4
Especificidade	característica da especificação do sistema abordar, apenas os requisitos necessários e de sua implementação gerar, apenas, as soluções especificadas |___|___|___|___|___| 0 1 2 3 4
Clareza	característica do sistema estar especificado, modelado e implementado da forma mais clara possível, isento de práticas que o tornem complexo e de difícil entendimento |___|___|___|___|___| 0 1 2 3 4
Concisão	característica do sistema ter sido especificado e modelado de forma concisa e implementado com a quantidade mínima de comandos |___|___|___|___|___| 0 1 2 3 4
Modularidade	característica do sistema ser projetado e implementado através de uma estrutura de módulos independentes e particionados logicamente |___|___|___|___|___| 0 1 2 3 4

Correção da representação	característica do sistema estar correto do ponto de vista do uso da linguagem de especificação adotada no que se refere a notação, semântica, sintaxe e formato de documentação |___|___|___|___|___| 0 1 2 3 4
Uniformidade de terminologia	característica do sistema estar documentado com uniformidade de notação e padronização de termos técnicos |___|___|___|___|___| 0 1 2 3 4
Uniformidade no grau de abstração	característica da documentação do sistema possuir um nível uniforme de detalhe, considerando-se um determinado estágio do desenvolvimento |___|___|___|___|___| 0 1 2 3 4
Completude da documentação	característica da documentação estar completa de acordo com os roteiros estabelecidos no plano de documentação do projeto |___|___|___|___|___| 0 1 2 3 4
Disponibilidade da documentação	característica do sistema ter sua documentação atualizada e pronta para uso quando necessário |___|___|___|___|___| 0 1 2 3 4
Estrutura	característica do sistema possuir um padrão definido de composição de suas partes, formando uma organização hierárquica de regras e/ou conhecimento |___|___|___|___|___| 0 1 2 3 4

Anexo IV

Listagens dos Resultados gerados pelo SPSS

1. Listagem do Teste de Wilcoxon para verificar a Hipótese 1

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.0

Mean Rank	Cases	
16,38	4	- Ranks (V1.0 LT ESPERADO)
20,96	36	+ Ranks (V1.0 GT ESPERADO)
	48	Ties (V1.0 EQ ESPERADO)
	--	
	88	Total

Z = -4,6305 2-Tailed P = ,0000

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.1

Mean Rank	Cases	
14,50	4	- Ranks (V1.1 LT ESPERADO)
17,34	29	+ Ranks (V1.1 GT ESPERADO)
	55	Ties (V1.1 EQ ESPERADO)
	--	
	88	Total

Z = -3,9756 2-Tailed P = ,0001

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.2

Mean Rank	Cases	
10,25	4	- Ranks (V1.2 LT ESPERADO)
15,21	24	+ Ranks (V1.2 GT ESPERADO)
	60	Ties (V1.2 EQ ESPERADO)
	--	
	88	Total

Z = -3,6890 2-Tailed P = ,0002

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test
ESPERADO
with V1.3

Mean Rank	Cases	
8,00	4	- Ranks (V1.3 LT ESPERADO)
10,53	15	+ Ranks (V1.3 GT ESPERADO)
	69	Ties (V1.3 EQ ESPERADO)
	--	
	88	Total

Z = -2,5353 2-Tailed P = ,0112

2. Listagem do Teste de Wilcoxon para verificar a Hipótese 2

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

V1.0
with V1.1

Mean Rank	Cases	
8,50	14	- Ranks (V1.1 LT V1.0)
11,33	3	+ Ranks (V1.1 GT V1.0)
	71	Ties (V1.1 EQ V1.0)
	--	
	88	Total

Z = -2,0119 2-Tailed P = ,0442

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

V1.0
with V1.2

Mean Rank	Cases	
12,05	20	- Ranks (V1.2 LT V1.0)
16,80	5	+ Ranks (V1.2 GT V1.0)
	63	Ties (V1.2 EQ V1.0)
	--	
	88	Total

Z = -2,1122 2-Tailed P = ,0347

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

V1.0
with V1.3

Mean Rank	Cases	
15,44	26	- Ranks (V1.3 LT V1.0)
15,88	4	+ Ranks (V1.3 GT V1.0)
	58	Ties (V1.3 EQ V1.0)
	--	
	88	Total

Z = -3,4760 2-Tailed P = ,0005

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

V1.1
with V1.2

Mean Rank	Cases	
7,43	7	- Ranks (V1.2 LT V1.1)
6,50	6	+ Ranks (V1.2 GT V1.1)
	75	Ties (V1.2 EQ V1.1)
	--	
	88	Total

Z = -,4543 2-Tailed P = ,6496

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

V1.1
with V1.3

Mean Rank	Cases	
12,55	19	- Ranks (V1.3 LT V1.1)
12,30	5	+ Ranks (V1.3 GT V1.1)
	64	Ties (V1.3 EQ V1.1)
	--	
	88	Total

Z = -2,5286 2-Tailed P = ,0115

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

V1.2
with V1.3

Mean Rank	Cases	
8,86	14	- Ranks (V1.3 LT V1.2)
9,67	3	+ Ranks (V1.3 GT V1.2)
	71	Ties (V1.3 EQ V1.2)
	--	
	88	Total

Z = -2,2486 2-Tailed P = ,0245

3. Listagem de testes realizados para analisar o comportamento do sistema

---------- Casos com resposta esperada na faixa 1 (44 casos) ----------

FILTER_$ Filter Status

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	44	100,0	100,0	100,0
	Total	44	100,0	100,0	

Valid cases 44 Missing cases 0

----- Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.0

Mean Rank	Cases	
,00	0	- Ranks (V1.0 LT ESPERADO)
14,00	27	+ Ranks (V1.0 GT ESPERADO)
	17	Ties (V1.0 EQ ESPERADO)
	44	Total

Z = -4,5407 2-Tailed P = ,0000

----- Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.1

Mean Rank	Cases	
,00	0	- Ranks (V1.1 LT ESPERADO)
10,50	20	+ Ranks (V1.1 GT ESPERADO)
	24	Ties (V1.1 EQ ESPERADO)
	44	Total

Z = -3,9199 2-Tailed P = ,0001

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.2

Mean Rank Cases

,00 0 - Ranks (V1.2 LT ESPERADO)
9,00 17 + Ranks (V1.2 GT ESPERADO)
27 Ties (V1.2 EQ ESPERADO)
--
44 Total

Z = -3,6214 2-Tailed P = ,0003

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.3

Mean Rank Cases

,00 0 - Ranks (V1.3 LT ESPERADO)
4,50 8 + Ranks (V1.3 GT ESPERADO)
36 Ties (V1.3 EQ ESPERADO)
--
44 Total

Z = -2,5205 2-Tailed P = ,0117

---------- **Casos com resposta esperada na faixa 2 (2 casos)** ----------

FILTER_$ Filter Status

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	2	100,0	100,0	100,0
	Total	2	100,0	100,0	

Valid cases 2 Missing cases 0

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.0

Mean Rank Cases

,00 0 - Ranks (V1.0 LT ESPERADO)
1,00 1 + Ranks (V1.0 GT ESPERADO)
1 Ties (V1.0 EQ ESPERADO)
-
2 Total

Z = -1,0000 2-Tailed P = ,3173

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.1

Mean Rank	Cases	
,00	0	- Ranks (V1.1 LT ESPERADO)
1,00	1	+ Ranks (V1.1 GT ESPERADO)
	1	Ties (V1.1 EQ ESPERADO)
	-	
	2	Total

Z = -1,0000 2-Tailed P = ,3173

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.2

Mean Rank	Cases	
,00	0	- Ranks (V1.2 LT ESPERADO)
1,00	1	+ Ranks (V1.2 GT ESPERADO)
	1	Ties (V1.2 EQ ESPERADO)
	-	
	2	Total

Z = -1,0000 2-Tailed P = ,3173

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.3

Mean Rank	Cases	
,00	0	- Ranks (V1.3 LT ESPERADO)
1,00	1	+ Ranks (V1.3 GT ESPERADO)
	1	Ties (V1.3 EQ ESPERADO)
	-	
	2	Total

Z = -1,0000 2-Tailed P = ,3173

---------- **Casos com resposta esperada na faixa 3 (14 casos)** ----------

FILTER_$ Filter Status

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	14	100,0	100,0	100,0
	Total	14	100,0	100,0	

Valid cases 14 Missing cases 0

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.0

Mean Rank	Cases	
7,50	2	- Ranks (V1.0 LT ESPERADO)
5,00	8	+ Ranks (V1.0 GT ESPERADO)
	4	Ties (V1.0 EQ ESPERADO)
	14	Total

Z = -1,2741 2-Tailed P = ,2026

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.1

Mean Rank	Cases	
7,50	2	- Ranks (V1.1 LT ESPERADO)
5,00	8	+ Ranks (V1.1 GT ESPERADO)
	4	Ties (V1.1 EQ ESPERADO)
	14	Total

Z = -1,2741 2-Tailed P = ,2026

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.2

Mean Rank	Cases	
6,00	2	- Ranks (V1.2 LT ESPERADO)
4,00	6	+ Ranks (V1.2 GT ESPERADO)
	6	Ties (V1.2 EQ ESPERADO)
	14	Total

Z = -,8402 2-Tailed P = ,4008

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.3

Mean Rank Cases

4,50 2 - Ranks (V1.3 LT ESPERADO)
4,50 6 + Ranks (V1.3 GT ESPERADO)
6 Ties (V1.3 EQ ESPERADO)
--
14 Total

Z = -1,2603 2-Tailed P = ,2076

---------- **Casos com resposta esperada na faixa 4 (28 casos)** ----------

FILTER_$ Filter Status

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	28	100,0	100,0	100,0
Total		28	100,0	100,0	

Valid cases 28 Missing cases 0

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.0

Mean Rank Cases

1,50 2 - Ranks (V1.0 LT ESPERADO)
,00 0 + Ranks (V1.0 GT ESPERADO)
26 Ties (V1.0 EQ ESPERADO)
--
28 Total

Z = -1,3416 2-Tailed P = ,1797

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.1

Mean Rank Cases

1,50 2 - Ranks (V1.1 LT ESPERADO)
,00 0 + Ranks (V1.1 GT ESPERADO)
26 Ties (V1.1 EQ ESPERADO)
--
28 Total

Z = -1,3416 2-Tailed P = ,1797

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.2

Mean Rank Cases

1,50 2 - Ranks (V1.2 LT ESPERADO)
,00 0 + Ranks (V1.2 GT ESPERADO)
26 Ties (V1.2 EQ ESPERADO)
--
28 Total

Z = -1,3416 2-Tailed P = ,1797

- - - - - Wilcoxon Matched-Pairs Signed-Ranks Test

ESPERADO
with V1.3

Mean Rank Cases

1,50 2 - Ranks (V1.3 LT ESPERADO)
,00 0 + Ranks (V1.3 GT ESPERADO)
26 Ties (V1.3 EQ ESPERADO)
--
28 Total

Z = -1,3416 2-Tailed P = ,1797

----------Determinação do número de casos (frequency) para cada versão ----------
---------- considerando resposta esperada na faixa 1 ----------

FILTER_$ Filter Status <=== esperado = 1 e v1.0 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	17	100,0	100,0	100,0
Total		17	100,0	100,0	

Valid cases 17 Missing cases 0

FILTER_$ Filter Status <=== esperado = 1 e v1.0 = 2

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	12	100,0	100,0	100,0
Total		12	100,0	100,0	

Valid cases 12 Missing cases 0

```
FILTER_$  Filter Status     <=== esperado = 1 e v1.0 = 3
                                                    Valid    Cum
Value Label            Value  Frequency  Percent  Percent  Percent

Selected                   1          6    100,0    100,0    100,0
                       -------  --------  -------
                 Total         6    100,0    100,0

Valid cases      6     Missing cases      0


FILTER_$  Filter Status     <=== esperado = 1 e v1.0 = 4
                                                    Valid    Cum
Value Label            Value  Frequency  Percent  Percent  Percent

Selected                   1          9    100,0    100,0    100,0
                       -------  --------  -------
                 Total         9    100,0    100,0

Valid cases      9     Missing cases      0

* ---------------------------------------------------------------------- */

FILTER_$  Filter Status       <=== esperado = 1 e v1.1 = 1
                                                    Valid    Cum
Value Label            Value  Frequency  Percent  Percent  Percent

Selected                   1         24    100,0    100,0    100,0
                       -------  --------  -------
                 Total        24    100,0    100,0

Valid cases     24     Missing cases      0

FILTER_$  Filter Status       <=== esperado = 1 e v1.1 = 2
                                                    Valid    Cum
Value Label            Value  Frequency  Percent  Percent  Percent

Selected                   1          9    100,0    100,0    100,0
                       -------  --------  -------
                 Total         9    100,0    100,0

Valid cases      9     Missing cases      0


FILTER_$  Filter Status       <=== esperado = 1 e v1.1 = 3

                                                    Valid    Cum
Value Label            Value  Frequency  Percent  Percent  Percent

Selected                   1          6    100,0    100,0    100,0
                       -------  --------  -------
                 Total         6    100,0    100,0

Valid cases      6     Missing cases      0
```

FILTER_$ Filter Status <=== esperado = 1 e v1.1 = 4

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	5	100,0	100,0	100,0
Total		5	100,0	100,0	

Valid cases 5 Missing cases 0

* ---------- */

FILTER_$ Filter Status <=== esperado = 1 e v1.2 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	27	100,0	100,0	100,0
Total		27	100,0	100,0	

Valid cases 27 Missing cases 0

FILTER_$ Filter Status <=== esperado = 1 e v1.2 = 2

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	4	100,0	100,0	100,0
Total		4	100,0	100,0	

Valid cases 4 Missing cases 0

FILTER_$ Filter Status <=== esperado = 1 e v1.2 = 3

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	7	100,0	100,0	100,0
Total		7	100,0	100,0	

Valid cases 7 Missing cases 0

FILTER_$ Filter Status <=== esperado = 1 e v1.2 = 4

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	6	100,0	100,0	100,0
Total		6	100,0	100,0	

Valid cases 6 Missing cases 0

* ---------- */

FILTER_$ Filter Status <=== esperado = 1 e v1.3 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	36	100,0	100,0	100,0
Total		36	100,0	100,0	

Valid cases 36 Missing cases 0

FILTER_$ Filter Status <=== esperado = 1 e v1.3 = 2

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	4	100,0	100,0	100,0
Total		4	100,0	100,0	

Valid cases 4 Missing cases 0

FILTER_$ Filter Status <=== esperado = 1 e v1.3 = 3

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	1	100,0	100,0	100,0
Total		1	100,0	100,0	

Valid cases 1 Missing cases 0

FILTER_$ Filter Status <=== esperado = 1 e v1.3 = 4

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	3	100,0	100,0	100,0
Total		3	100,0	100,0	

Valid cases 3 Missing cases 0

----------Determinação do número de casos (frequency) para cada versão ----------
---------- considerando resposta esperada na faixa 3 ----------

FILTER_$ Filter Status <=== esperado = 3 e v1.0 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	1	100,0	100,0	100,0
Total		1	100,0	100,0	

Valid cases 1 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.0 = 2

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	1	100,0	100,0	100,0
Total		1	100,0	100,0	

Valid cases 1 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.0 = 3

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	4	100,0	100,0	100,0
Total		4	100,0	100,0	

Valid cases 4 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.0 = 4

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	8	100,0	100,0	100,0
Total		8	100,0	100,0	

Valid cases 8 Missing cases 0

* ---------- */

FILTER_$ Filter Status <=== esperado = 3 e v1.1 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	1	100,0	100,0	100,0
Total		1	100,0	100,0	

Valid cases 1 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.1 = 2

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	1	100,0	100,0	100,0
Total		1	100,0	100,0	

Valid cases 1 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.1 = 3

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	4	100,0	100,0	100,0
Total		4	100,0	100,0	

Valid cases 4 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.1 = 4

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	8	100,0	100,0	100,0
Total		8	100,0	100,0	

Valid cases 8 Missing cases 0

* ---------- */

FILTER_$ Filter Status <=== esperado = 3 e v1.2 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	1	100,0	100,0	100,0
Total		1	100,0	100,0	

Valid cases 1 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.2 = 2

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	1	100,0	100,0	100,0
Total		1	100,0	100,0	

Valid cases 1 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.2 = 3

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	6	100,0	100,0	100,0
Total		6	100,0	100,0	

Valid cases 6 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.2 = 4

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	6	100,0	100,0	100,0
Total		6	100,0	100,0	

Valid cases 6 Missing cases 0

* ---------- */

FILTER_$ Filter Status <=== esperado = 3 e v1.3 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Total		0	100,0	100,0	

Valid cases 0 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.3 = 2

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	2	100,0	100,0	100,0
Total		2	100,0	100,0	

Valid cases 2 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.3 = 3

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	6	100,0	100,0	100,0
Total		6	100,0	100,0	

Valid cases 6 Missing cases 0

FILTER_$ Filter Status <=== esperado = 3 e v1.3 = 4

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	6	100,0	100,0	100,0
Total		6	100,0	100,0	

Valid cases 6 Missing cases 0

----------Determinação do número de casos (frequency) para cada versão ----------
---------- considerando resposta esperada na faixa 4 ----------

FILTER_$ Filter Status <=== esperado = 4 e v1.0 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Total	0	0	100,0	100,0	

Valid cases 0 Missing cases 0

FILTER_$ Filter Status <=== esperado = 4 e v1.0 = 2

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Total	0	0	100,0	100,0	

Valid cases 0 Missing cases 0

FILTER_$ Filter Status <=== esperado = 4 e v1.0 = 3

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	2	100,0	100,0	100,0
Total		2	100,0	100,0	

Valid cases 2 Missing cases 0

FILTER_$ Filter Status <=== esperado = 4 e v1.0 = 4

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Selected	1	26	100,0	100,0	100,0
Total		26	100,0	100,0	

Valid cases 26 Missing cases 0

* ---------- */

FILTER_$ Filter Status <=== esperado = 4 e v1.1 = 1

Value Label	Value	Frequency	Percent	Valid Percent	Cum Percent
Total	0	0	100,0	100,0	

Valid cases 0 Missing cases 0

```
FILTER_$ Filter Status          <=== esperado = 4 e v1.1 = 2
                                                    Valid    Cum
Value Label             Value  Frequency  Percent  Percent  Percent

                                -------  -------  -------
                  Total         0        100,0    100,0

Valid cases      0     Missing cases      0


FILTER_$ Filter Status          <=== esperado = 4 e v1.1 = 3
                                                    Valid    Cum
Value Label             Value  Frequency  Percent  Percent  Percent

Selected                1       2        100,0    100,0    100,0
                        -------  -------  -------
                  Total         2        100,0    100,0

Valid cases      2     Missing cases      0


FILTER_$ Filter Status          <=== esperado = 4 e v1.1 = 4
                                                    Valid    Cum
Value Label             Value  Frequency  Percent  Percent  Percent

Selected                1      26        100,0    100,0    100,0
                        -------  -------  -------
                  Total        26        100,0    100,0

Valid cases      26    Missing cases      0
* ---------------------------------------------------------------------- */

FILTER_$ Filter Status          <=== esperado = 4 e v1.2 = 1
                                                    Valid    Cum
Value Label             Value  Frequency  Percent  Percent  Percent

                                -------  -------  -------
                  Total         0        100,0    100,0

Valid cases      0     Missing cases      0


FILTER_$ Filter Status          <=== esperado = 4 e v1.2 = 2
                                                    Valid    Cum
Value Label             Value  Frequency  Percent  Percent  Percent

                                -------  -------  -------
                  Total         0        100,0    100,0

Valid cases      0     Missing cases      0


FILTER_$ Filter Status          <=== esperado = 4 e v1.2 = 3
                                                    Valid    Cum
```

```
Value Label          Value  Frequency  Percent  Valid Percent  Cum Percent

Selected                 1          2    100,0          100,0        100,0
                      -------    -------  -------
                Total    2    100,0    100,0

Valid cases      2    Missing cases      0
FILTER_$  Filter Status          <=== esperado = 4 e v1.2 = 4
                                                    Valid    Cum
Value Label          Value  Frequency  Percent  Valid Percent  Cum Percent

Selected                 1         26    100,0          100,0        100,0
                      -------    -------  -------
                Total   26    100,0    100,0

Valid cases     26    Missing cases      0
* ------------------------------------------------------------------------ */

FILTER_$  Filter Status          <=== esperado = 4 e v1.3 = 1
                                                    Valid    Cum
Value Label          Value  Frequency  Percent  Valid Percent  Cum Percent

                      -------    -------  -------
                Total    0    100,0    100,0

Valid cases      0    Missing cases      0


FILTER_$  Filter Status          <=== esperado = 4 e v1.3 = 2
                                                    Valid    Cum
Value Label          Value  Frequency  Percent  Valid Percent  Cum Percent

                      -------    -------  -------
                Total    0    100,0    100,0

Valid cases      0    Missing cases      0


FILTER_$  Filter Status          <=== esperado = 4 e v1.3 = 3
                                                    Valid    Cum
Value Label          Value  Frequency  Percent  Valid Percent  Cum Percent

Selected                 1          2    100,0          100,0        100,0
                      -------    -------  -------
                Total    2    100,0    100,0

Valid cases      2    Missing cases      0


FILTER_$  Filter Status          <=== esperado = 4 e v1.3 = 4
                                                    Valid    Cum
Value Label          Value  Frequency  Percent  Valid Percent  Cum Percent

Selected                 1         26    100,0          100,0        100,0
                      -------    -------  -------
                Total   26    100,0    100,0

Valid cases     26    Missing cases      0
```

Anexo V
Formulários para Testes

1.Formulário para Registro de Casos de Testes com Análise de Sensibilidade

PROJETO SEC - Casos de Teste

Responsável pelo Caso de Teste: XXXXX
Data: XX/XX/XX

CASO BÁSICO	VARIAÇÃO 1	VARIAÇÃO 2	VARIAÇÃO 3	VARIAÇÃO 4
	RESULTADO	**ESPERADO**		

2. Formulário para Registro de Casos do Teste em Campo

FBC - FUNDAÇÃO BAHIANA DE CARDIOLOGIA
SEC - SISTEMA ESPECIALISTA EM CARDIOLOGIA
VALIDAÇÃO COM CASOS REAIS

Versão : _____
Nome do responsável pela validação : _____
Data : _____

Nome do paciente :
Sexo :
Idade :
Dados :

Diagnóstico do Cardiologista :

( ) **Liberar (muito baixa possibilidade de evento coronariano agudo)**
( ) **Liberar com retorno (baixa possibilidade de evento coronariano agudo)**
( ) **Observação (possibilidade intermediária de evento coronariano agudo)**
( ) **Enviar a Unidade Coronariana (alta possibilidade de evento coronariano agudo)**

Diagnóstico do SEC :

( ) **Faixa 1 (muito baixa possibilidade de evento coronariano agudo)**
( ) **Faixa 2 (baixa possibilidade de evento coronariano agudo)**
( ) **Faixa 3 (possibilidade intermediária de evento coronariano agudo)**
( ) **Faixa 4 (alta possibilidade de evento coronariano agudo. Enviar a Unidade Coronariana)**

Dados de acompanhamento do paciente :

3. Formulário para Teste de Turing

PROJETO SEC - Casos de Testes

Caso de Teste: _____

Identificação do paciente:

Idade: _____

Sexo: ( ) M ( ) F

Investigação da ida do paciente ao posto de saúde:

Dor:

( ) Dor na região precordial ou médio-esternal
( ) Dor na região epigástrica
( ) Dor na mandíbula
( ) Dor no braço esquerdo
( ) Dor no dorso
( ) Dor no hemitorax direito

Característica da dor:

( ) Dor em aperto ou opressão
( ) Dor em peso ou queimor
( ) Dor em pontada
( ) Dor dependente da posição do paciente
( ) Dor ventilatório dependente
( ) Dor melhora ou desaparece com o uso de nitrato sublingual

Intensidade da dor:

( ) Leve
( ) Moderada
( ) Severa

Duração da dor: _____

Outros sintomas:

( ) Febre
( ) Sudorese
( ) Palpitações Associadas
( ) Náuseas e vômitos
( ) Dispnéia em repouso
( ) Dispnéia aos esforços

Outros:

_____
_____
_____
_____
_____

Fatores Predisponentes:

( ) Tabagismo
( ) Hipertensão Arterial
( ) Dislipidemia
( ) Diabete
( ) Obesidade
( ) Stress
( ) Pós-menopausa
( ) Episódios repetidos de dor torácica
( ) Antecedentes de problemas digestivos
( ) Evento coronário prévio
( ) Prolapso da valvula mitral
( ) Dor / desconforto precordial há menos de 2 semanas
( ) Cardiopatia isquêmica na família com idade inferior a 55 anos
( ) Diabete na família

Outros:

_____
_____
_____
_____
_____

Exame físico:

( ) Facies de sofrimento agudo
( ) Palidez
( ) Pele viscosa
( ) Vômitos
( ) Dor à palpação no local referido
( ) Pulso Filiforme
( ) Pulso Arrítmico
( ) Atrito Pericárdico
( ) Crepitos de base
( ) Batimento cardíaco com ritmo de galope
( ) Existe diferença considerável de amplitude entre os membros superiores e inferiores

Temperatura: _____ °C

Frequência Cardíaca: _____ bmp

Pressão arterial: _____ mmHg

Outros:

_____
_____
_____
_____
_____

Resultado do ECG:

( ) Complexo QRS com padrão Q ou QS
( ) Onda T negativa ou isquêmica
( ) Segmento ST infradesnivelado
( ) Segmento ST supradesnivelado
( ) Apresenta bloqueio de ramo esquerdo

Outros:

_____
_____
_____
_____
_____

Caso de Teste: _____ _____

CONCLUSÃO

( ) O paciente apresenta muito baixa possibilidade de estar com Evento Coronariano Agudo.

( ) O paciente apresenta baixa possibilidade de estar com Evento Coronariano Agudo.

( ) O paciente apresenta possibilidade intermediária de estar com Evento Coronariano Agudo.

( ) O paciente apresenta alta possibilidade de estar com Evento Coronariano Agudo devendo ser imediatamente encaminhado para uma Unidade Coronariana.