**Universidade Federal do Ceará**
**Centro de Ciências**
**Departamento de Computação**
**Mestrado e Doutorado em Ciência da Computação**

**Maria de Fátima Costa de Souza**

# Um Ambiente de Apoio à Seleção de Software Educativo

Fortaleza - CE, Brasil

Março de 2006

**Universidade Federal do Ceará**
**Centro de Ciências**
**Departamento de Computação**
**Mestrado e Doutorado em Ciência da Computação**

*Dissertação de Mestrado*

# Um Ambiente de Apoio à Seleção de Software Educativo

**Maria de Fátima Costa de Souza**

Orientador:
Prof. Dr. Mauro Cavalcante Pequeno

Co-orientador:
Prof. Dr. José Aires de Castro Filho

Fortaleza - CE, Brasil

Março de 2006

Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Mestrado e Doutorado em Ciência da Computação

**Maria de Fátima Costa de Souza**

# Um Ambiente de Apoio à Seleção de Software Educativo

Dissertação apresentada como requisito para a obtenção do Grau de Mestre em Ciência da Computação pela Universidade Federal do Ceará.

# Agradecimentos

Primeiramente gostaria de agradecer à DEUS por ter me fortalecido diante de tantas dificuldade, permitindo a conclusão deste trabalho.

Ao meus orientadores Prof. Dr. Mauro Cavalcante Pequeno e Prof. Dr. José Aires de Castro Filho pela dedicação, paciência e acima de tudo confiança em meu trabalho.

Agradeço aos meus queridos pais, Iraci e Expedito, que diante de tantas dificuldades, nunca se mostraram fracos, injetendo em mim a força que necessitava para concluir este trabalho.

Agradeço a minha querida mãe Iraci, que partiu antes que eu concluisse essa jornada, pelo exemplo e amor que me proporcionou durante toda sua vida.

Aos meus queridos sogros, Ivan e Cícero, que nunca mediram esforços para me ajudar.

Aos meus irmãos e irmãs que me ajudaram em momentos bastante difíceis.

A minha prima Juscilene, por sua grande dedicação.

A Alexsandra, secretária da UFC Virtual, pela dedicação e eficiência em sempre solucionar os problemas.

Ao Orley, secretário do mestrado, pela grande paciência e dedicação dispendidos a todos os alunos do mestrado.

A todos os professores que compoem o mestrado e doutorado em ciência da computação.

Agradeço ainda ao meu querido Cidcley pelo incentivo, confiança e paciência dispendidos a mim ao longo deste trabalho.

À FUNCAP por ter financiado meus estudos de Mestrado e consequentemente possibilitado o desenvolvimento dessa pesquisa.

*“A meta final de qualquer pesquisa não é a objetividade, mas a verdade.”*

Helene Deutsch

# Resumo

Atualmente, em virtude do intenso desenvolvimento de produtos de software educativos (SE), a tarefa de seleção desses produtos, pelos professores, para uso em sala de aula, tem se tornado cada vez mais difícil. Argumentamos nesse trabalho, que mesmo existindo inúmeras estratégias de avaliação elas não são efetivamente aplicadas para realizar uma seleção baseada na comparação das características dos produtos de software avaliados. Sendo assim, apresentamos uma nova metodologia para avaliação de software educativo que permite a seleção de software com características funcionais semelhantes, utilizando métricas definidas pelo próprio avaliador e baseada na Teoria dos Campos Conceituais com a utilização e formalização da técnica de Pontos de Casos de Uso. Essa metodologia dará suporte a avaliação comparativa de SE, de forma a considerar de maneira simultânea tanto aspectos relativos a funcionalidade do produto quanto aspectos pedagógicos/cognitivos. Para isso, procuramos quantificar as avaliações, através da utilização do ajuste de métricas previamente estabelecidas, juntamente com a aplicação da técnica de Pontos de Casos de Uso, no intuito de conseguir gerar uma pontuação para cada software, permitindo assim a realização de uma seleção mais refinada de um SE, dando condições ao avaliador de escolher, de forma segura, o software que mais se adeque às suas necessidades. Além disso, no intuito de facilitar o processo de utilização dessa metodologia, propomos ainda uma ferramenta que irá automatizar todo o processo de avaliação, tornando a utilização desta metodologia acessível a qualquer usuário.

# Abstract

Nowadays, because of the intense development of educational software (ES), the task of selecting these tools by professors for their use in the classroom has become harder. We argue in this paper that, even with the large number of evaluation strategies available, they are not effectively applied in the selection based on the comparison of the characteristics of such tools. Therefore, we present a new methodology for the evaluation of educational software that allows the selection of software with similar functional characteristics, using metrics defined by the evaluator himself/herself and based on the Conceptual Fields Theory with the use and formalization of the Use Case Points technique. This methodology will enable the comparative evaluation of ES, by taking into consideration simultaneously both aspects related to the product functioning as well as cognitive/pedagogical aspects. In order to achieve that, we quantify the evaluation through the adjustment of previously chosen metrics, along with the application of the Use Case Points technique in the generation of a score for each piece of software, in such a way to allow a more refined evaluation of a ES, providing to the evaluator the necessary set of conditions required for choosing the piece of software that is more adequate to his/her needs. In addition, in order to facilitate the use of the methodology, we propose a tool that will make the whole evaluation process automatic, making the application of our methodology accessible for everyone.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# 1 Introdução

## 1.1 Contexto

Mais do que obter um bom desempenho em exercícios pré-definidos, ou a memorização de fórmulas, um dos objetivos centrais do ensino, em particular o de matemática, é conseguir que os alunos desenvolvam uma compreensão dos conceitos. Através dessa compreensão eles serão capazes de conseguir o que se denomina como pensamento matemático avançado (PINTO, 2001). Para TALL (1988), esse pensamento é construído ao longo dos anos através de experiências de todas as espécies, mudando quando os indivíduos são confrontados com novos estimulo. A fim de atingir essa forma de pensamento, a seqüência de atividades de aprendizagem usadas devem promover a interação de uma variedade de processos mentais. A inclusão de atividades do tipo laboratorial é uma das vertentes educativas fundamentais, como o modo de conseguir qualidade na aprendizagem.

Os laboratórios de matemática constituem um meio privilegiado de permitir explorações de conceitos matemáticos (PINTO, 2001). É nesse contexto que os computadores têm um papel fundamental a desempenhar, possibilitando a passagem de experiências gráficas e numéricas iniciais para construções analíticas. O computador como ferramenta pode ser usado ainda, semelhante a um microscópio para o biólogo, que se for focada corretamente, poderá apresentar uma perspectiva surpreendente, conduzindo a novas idéias e ao reconhecimento de relações até então ignoradas. Embora no caso do biólogo, estas idéias e relações sejam de sua própria autoria, no caso do aluno, elas serão constituídas de fatos matemáticos, já existentes, muito embora, sejam novas para esse aluno ou para a turma. Na busca de alternativas didáticas, passou-se a ver o computador como um forte recurso didático que se pode usar em salas de aula como elemento de apoio para o ensino. Muito embora ainda não estejam disponíveis para a maioria das escolas, eles já começam a integrar muitas experiências educacionais (OLIVEIRA; KLUPPEL, 1999).

Tem-se observado, de forma cada vez mais intensa, o lançamento no mercado de softwares que, segundo seus fabricantes, poderiam auxiliar o trabalho de professores e facilitar a

aprendizagem dos alunos. No entanto, grande parte desses programas é de baixa qualidade sob os aspectos pedagógicos, o que os torna sem utilidade para uso em ambientes de ensino-aprendizagem (TEODORO; FREITAS, 1992). Desta forma, é fundamental que se faça a avaliação sistemática da qualidade e dos efeitos de tais softwares. Porém, tal procedimento não ocorre com freqüência, visto que muitas instituições de ensino adquirem programas que são utilizados pelos alunos sem uma avaliação prévia (SILVA, 2002).

Avaliar um Software Educativo, significa analisar as suas características e implicações para o uso educacional. No processo de avaliação de software é importante observar a natureza do mesmo e aspectos técnicos. Em geral, não se faz referência a uma concepção de aprendizagem que norteie a aprendizagem mediada pelo software.

Com relação à natureza do software, VALENTE (1999) classifica software educativos, de acordo com seus objetivos pedagógicos, em: tutoriais, aplicativos, programação, exercícios e prática, multimídia e Internet, simulação, modelagem e jogos. Para OLIVEIRA, GOMES e BORGES NETO (2001), os softwares podem ser de caráter mais fechado ou não, isto é, o software fechado não permite que o aluno verifique o processo, mas somente o produto final.

Mesmo com o uso de softwares, o papel do professor continua fundamental para a aprendizagem dos alunos e assume aspectos diversificados, como a valorização das pequenas descobertas dos alunos. Os alunos necessitam que o professor ajude a sistematizar os elementos obtidos experimentalmente como, por exemplo, chamando a atenção para regularidades. Cabe ao professor escolher uma seqüência de ensino que torne os conceitos a aprender mais facilmente compreensíveis, bem como intervir no sentido de chamar atenção para os aspectos mais abstratos, que não são diretamente observáveis apenas através da experimentação (PINTO, 2001).

Para que o professor consiga tirar proveito do ferramental tecnológico disponível atualmente no mercado, não basta que aprenda a usar o computador, é necessário retirar do seu uso aquilo que realmente interessa aos objetivos educacionais. Dessa forma, as seguintes questões são de suma importância: Como selecionar diferentes software disponíveis? Como utilizá-los e melhor aproveitá-los para ensinar Matemática? Como integrá-los com os conteúdos existentes?

A seleção de softwares deve ser feita pelo professor que está em contato direto com o processo de ensino e aprendizagem, pois é ele quem vai identificar as dificuldades dos alunos, por meio da análise de suas ações, e vai propor o uso de materiais os mais adequados a criar as situações favoráveis à aprendizagem dos conceitos mal compreendidos (OLIVEIRA; GOMES; BORGES NETO, 2001).

É fundamental que seja feita uma correta escolha de um software em função dos objetivos que se pretende atingir, distinguindo os que são voltados mais para um trabalho dirigido de mero exercício e prática, daqueles que levam o aluno a interagir com o programa, de forma a explorar hipóteses. Para que essa escolha seja possível, os softwares devem ser previamente avaliados.

Entretanto, as técnicas de avaliação de software educativos disponíveis atualmente, seguem, tradicionalmente, grades de categorias oriundas do campo da engenharia de software que focalizam parâmetros gerais relativos à qualidade da interface, à coerência de apresentação dos conceitos e aos aspectos ergonômicos gerais dos sistemas. Esta avaliação é feita a partir da aplicação de tabelas de critérios nas quais aspectos como: consistência da representação, usabilidade, qualidade da interface, qualidade do *feedback*, são considerados segundo uma escala de três ou quatro níveis (GOMES et al., 2002).

A literatura sobre avaliação de software educativo é abundante em adaptações de tabelas que ora se adaptam ao tipo de software (independentemente do conteúdo veiculado), ora adaptam-se ao tipo de ferramenta (software ou site). Esta literatura busca pontuar aspectos importantes na análise de um software educativo como: idioma, conteúdos abordados, público alvo, documentação (ficha técnica clara e objetiva, manual do professor com sugestões para o uso, ajuda on-line); aspectos pedagógicos (facilidade no acesso às informações, adequação à faixa etária, clareza nas informações, tipo de exercícios); interface (facilidade de uso, interatividade com o usuário, qualidade de áudio, gráficos e animação, recursos de avançar e recuar, adaptação do usuário); conteúdos (fidelidade ao objeto, coerência de apresentação do conteúdo, correção dos exercícios, organização dos conteúdos, promoção da criatividade e motivação dos usuários); feedback (forma deste e qualidade da motivação); aspectos técnicos (instalação, manipulação, apresentação visual e controle dos comandos); avaliação (forma de avaliação, tempo destinado às respostas, forma de correção e de orientação) e aspectos gerais (alcançar os objetivos propostos, contribuir para a aprendizagem dos conteúdos apresentados, preço compatível) como entendem GOMES et al. (2002).

Como podemos observar, as metodologias para avaliação de software educativo são basicamente voltadas à verificação da presença de determinados requisitos nos software avaliados. Em alguns poucos casos há a utilização de algum tipo de pontuação simples onde esses pontos são levantados basicamente a partir de observações superficiais e informais. Por conta dessa informalidade, podemos concluir que essas metodologias não são tão eficazes quando o objetivo principal é comparar software de mesmo domínio a partir de seus requisitos não funcionais

## 1.2 Motivação

As metodologias de avaliação de SE apresentadas na literatura, em sua grande maioria, levam em consideração aspectos individuais de um software, como aspectos pedagógicos ou técnicos, no entanto essas metodologias se tornam limitadas quando o professor possue uma relação de software que abordam o mesmo conteúdo e entre eles tenha que selecionar o que melhor lhe adequar.

Desse modo, as metodologias existentes nos respondem apenas perguntas do tipo:

- Esse software possui uma interface amigável?
- Esse software é fácil de usar?
- Esse software proporciona situações em que são fornecidos significados aos conceitos matemáticos trabalhados?

Se por exemplo, formos comparar três softwares que tenham a mesma área de aplicação, utilizando as metodologias atuais, é possível que todos venham a ter tanto uma interface amigável como ser de fácil uso. Dessa forma, podemos chegar à absurda conclusão de que os softwares são iguais.

O fato de existir uma grande variedade de SE disponíveis atualmente que podem ser utilizados para trabalhar um mesmo conteúdo, torna a tarefa de avaliação bastante difícil, principalmente se considerarmos as técnicas de avaliação propostas atualmente na literatura (pontuações que variam de 0 a 4). De fato, essas técnicas têm como ênfase a análise tanto de aspectos cognitivos como de aspectos de usabilidade, normalmente utilizando métricas simplificadas, de forma a categorizar um determinado software ou mostrar que certos aspectos estão ou não presentes. Desse modo, em virtude da expressividade limitada dessas métricas, essas metodologias acabam não sendo eficazes quando a tarefa que está sendo realizada é a comparação de produtos de software [1].

Neste trabalho, propomos uma nova metodologia de apoio à avaliação de SE. Nossa abordagem se baseia na idéia de formalizar tanto aspectos pedagógicos como técnicos de forma a possibilitar a realização de uma avaliação comparativa precisa entre SE. Nossa metodologia propõe a adoção e a especificação de métricas para aspectos relevantes definidas pelo próprio professor a serem aferidos nos SE que estão sendo avaliados.

[1] Produto de software é um termo utilizado em Engenharia de Software para identificar um software que possa ser vendido/fornecido a um cliente.

Para alcançar esses objetivos, fazemos uso neste trabalho, da técnica de Pontos de Casos de Uso, proposta por Karner (KARNER, 1993), que é largamente utilizada para estimar esforço de desenvolvimento de software, juntamente com a Teoria dos Campos Conceituais de Vergnaud (VERGNAUD, 1990). A utilização conjunta dessa técnica com a teoria dos campos conceituais permite a captura dos requisitos funcionais, não funcionais, e dos fatores pedagógicos, respectivamente, levando em consideração uma perspectiva pedagógica construtivista.

## 1.3 Objetivos

Neste trabalho, temos como objetivo principal, apresentar uma nova metodologia de apóio à seleção de produtos de SE de matemática, em que visamos os seguintes aspectos:

1. Tratar avaliação de SE, considerando tanto aspectos funcionais, não-funcionais e de domínio (pedagógicos/cognitivos) de forma integrada;
2. Utilizar técnicas de Engenharia de Software para realizar a elicitação, especificação e aferição de requisitos no contexto de avaliação de software;
3. Aplicar a Teoria dos Campos Conceituais de Vergnaud na avaliação de SE, no sentido de que todos os seus aspectos sejam considerados de forma relacionada.
4. Por fim, criar uma ferramenta que seja capaz de conciliar termos da engenharia de software com a Teoria dos Campos Conceituais, tornando-a acessível a qualquer usuário.

Desse modo, essa nova metodologia tem o intuito de gerar valores comparativos que permitirão uma seleção entre diferentes SE de matemática. Para isso, consideraremos simultaneamente requisitos funcionais, não-funcionais e de domínio (pedagógico/cognitivo).

Os requisitos funcionais serão especificados através de casos de uso, os requisitos de domínio serão avaliados mediante a teoria dos campos conceituais de Vergnaud e para os requisitos não-funcionais definiremos métricas a fim de avaliar a importância de cada um dos requisitos.

Propomos assim, uma estratégia de geração de valores para cada um desses tipos de requisitos de forma a aplicar diretamente a técnica de pontos de caso de uso para a obtenção de valores a serem utilizados na tarefa de seleção do software.

## 1.4 Organização da Dissertação

Esse trabalho está organizado em oito capítulos. No capítulo 1, introduzimos de maneira sucinta o tema que serviu de motivação para o desenvolvimento deste trabalho, bem como a definição de seus objetivos.

No capítulo 2, apresentamos como são classificados os software educativo e as metodologias existentes para avaliá-los.

No capítulo 3, apresentamos a Teoria dos Campos Conceituais como um dos pilares de nosso trabalho, bem como seu conceito de "conceito", situação, invariante e esquemas.

No capítulo 4, apresentamos a definição de requisitos de software, juntamente com a definição de Casos de Uso, além de sua representação através de Diagramas. Ainda neste capítulo abordamos algumas técnicas de estimativa de software, dando ênfase aos Pontos de Casos de Uso, seu conceito e sua utilização.

No capítulo 5, apresentamos a nossa proposta, que é a metodologia de avaliação comparativa de software educativo, com suas devidas especificações de Casos de Uso, bem como seus cálculos.

No capítulo 6, fazemos a validação de nossa proposta, através da análise de dois softwares que apresentam características funcionais e não funcionais bastante semelhantes, que a priori seriam de difícil seleção pelo avaliador.

No capítulo 7, apresentamos uma ferramenta que irá facilitar o uso de nossa metodologia, tornando-a acessível a qualquer usuário.

Por fim, no capítulo 8, apresentamos algumas considerações sobre a importância da metodologia aplicada numa avaliação de SE, pois a partir dela o avaliador poderá selecionar os requisitos que mais lhe interessam, permitindo a realização de uma seleção mais refinada. Ainda no capítulo 8, mencionamos as nossas contribuições com o desenvolvimento deste trabalho, além de citarmos as nossas pretenções quanto aos trabalhos futuros. E para referendar a nossa pesquisa citamos as nossas publicações durante esse período.

# 2 Metodologias Atuais Para Avaliação de Software Educativo

## 2.1 Introdução

É grande a proliferação dos mais diversos tipos e aplicações de software educativos, muitos de qualidade falaciosa no mercado, que em nada vem acrescentar o desenvolvimento do aluno.

É visando a qualidade desses diversos tipos de software que estão sendo produzidos, que o termo avaliar passou a ser criteriosamente utilizado. Avaliar um software, nada mais é que verificar se esses programas são capazes de agregar valores ao aprendizado do aluno, pois todo software educativo deve refletir necessariamente uma concepção de ensino e aprendizagem, resultante de uma visão filosófica da relação sujeito-objeto.

Tendo em vista a atual diversidade de materiais educativos informatizados, cabe aos responsáveis pela educação o questionamento sobre a aplicabilidade desses programas como ferramentas didático-pedagógicas bem como o seu nível de qualidade.

OLIVEIRA, COSTA e MOREIRA (2001) apresentam duas formas de avaliação de software educativo: A formativa e a objetiva. A primeira é um processo que acompanha a utilização do SE em um ambiente real de aprendizagem, em que os alunos interagem com seu objeto de conhecimento. A segunda avaliação prevê a utilização de listas de critérios, disponibilizada por diferentes autores.

A avaliação formativa é a forma de avaliação em que a preocupação central reside em coletar dados para reorientação do processo de ensino-aprendizagem. Trata-se de uma “bússola orientadora” do processo de ensino-aprendizagem que não deve exprimir-se através de uma nota, mas sim por meio de comentários.

A avaliação objetiva (somativa), como o próprio nome indica, tem como objetivo representar um sumário, uma apresentação concentrada de resultados obtidos numa situação educativa. Pretende-se traduzir, de uma forma quantificada, a distância em que ficou de uma meta que se

arbitrou ser importante atingir. Segundo BLAYA (2004), Essa avaliação tem lugar em momentos específicos, ao longo de um curso, como por exemplo, no final de um ano letivo.

Sendo assim, as formas adotadas de avaliação devem ser realizadas de forma cautelosa, pois a escolha feita indica a filosofia que orientará o processo de ensino-aprendizagem.

Diante dessa difícil escolha, apresentamos a seguir os tipos e as características de software educativo que podem ser utilizados dentro do contexto escolar, como forma de auxiliar o educador a alcançar o seu objetivo, levando em consideração seus princípios pedagógicos.

## 2.2 Tipos e Características de Software Educativo

O SE é uma ferramenta que auxilia e apóia o professor no processo de ensino e aprendizagem buscando favorecer a aquisição de conhecimento pelo aluno. Esse favorecimento é o que distingue o SE de programas produzidos no mercado, com outras finalidades como empresarial, gerencial. administrativa etc. No entanto, para que o uso dessa ferramenta seja eficiente é necessário que se faça uma seleção de fatores e conteúdos que se deseja trabalhar com os alunos.

Sendo assim, apresentamos aqui, a partir dos trabalhos de VALENTE (1999), OLIVEIRA, COSTA e MOREIRA (2001), entre outros, algumas modalidades e características de SE, levando em consideração a função que os mesmos desempenham dentro do contexto de ensino-aprendizagem.

### 2.2.1 Tutorial

Para OLIVEIRA, COSTA e MOREIRA (2001), os tutorias apresentam essencialmente informações que que são transmitidas em um diálogo entre o usuário (aprendiz) e o computador, tendo como característica a apresentação de informações, resposta a uma ou mais perguntas, ou ainda a solução de problemas. Estas informações apresentadas deverão motivar e estimular o aprendiz, para que haja comprometimento em alguma ação relacionada com a informação.

Os programas tutoriais podem introduzir conceitos novos, apresentar habilidades, pretender a aquisição de conceitos, princípios e ou generalizações através da transmissão de determinado conteúdo ou da proposição de atividades que verifiquem a aquisição deste conteúdo. Servem como apoio ou reforço para aulas, para preparação ou revisão de atividades, entre outros aspectos.

Para VALENTE (1999), a vantagem dos tutoriais é o fato do computador poder apresentar

o material com outras características que não são permitidas no papel como: animação, som e a manutenção do controle do desempenho do aprendiz, facilitando o processo de administração das lições e possíveis programas de remediação. Além destas vantagens, os programas tutoriais são bastante usados pelo fato de permitirem a introdução do computador na escola sem provocar muita mudança isto é, a versão computadorizada do que já acontece na sala de aula, onde o professor necessita de pouquíssimo treino para o seu uso e o aluno já sabe qual é o seu papel como aprendiz.

Esses softwares têm como característica a existência de recursos motivacionais; controle da sequenciação do programa pelo usuário; mensagens de erro, com o intuito de conduzir o aluno a resposta correta ou desejada, dentre outras.

### 2.2.2 Exercícios e Práticas

Os programas de exercício e prática são a forma mais tradicional em que os computadores têm sido utilizado em educação. Visa a aquisição de uma habilidade ou a aplicação de um conteúdo já conhecido pelo aluno, mas não inteiramente dominado. Podem suplementar o ensino em sala de aula, aumentar e/ou automatizar habilidades básicas.

Em geral, são programas que tem como intuito, reforçar fatos e conhecimentos que são analisados em uma aula expositiva ou em um laboratório, através de perguntas e respostas (OLIVEIRA; COSTA; MOREIRA, 2001). Podemos ainda citar como características dessa modalidade, a utilização do feedback positivo para melhorar o desempenho do aluno, o uso da seleção randômica de problemas por parte dos mesmos, a repetição de exercícios como maneira de atingir os objetivos determinados no programa, além da detecção rápida de respostas erradas, reduzindo assim a possibilidade de reforço em procedimentos errôneos.

A vantagem deste tipo de programa é o fato do professor dispor de uma infinidade de exercícios que o aprendiz pode resolver de acordo com o seu grau de conhecimento e interesse. Se o software, além de apresentar o exercício, coletar as respostas de modo a verificar o desempenho do aprendiz, então o professor terá à sua disposição um dado importante sobre como o material visto em classe está sendo absorvido. Entretanto, para alguns professores, este dado não é suficiente porque é muito difícil para o software detectar o porquê do acerto ou erro do aluno. A forma como o assunto está sendo assimilado exige um conhecimento muito mais amplo do que o número de acertos e erros dos aprendizes. Portanto, a idéia de que os programas de exercício-e-prática aliviam a tediosa tarefa dos professores corrigirem os testes ou as avaliações não é totalmente verdadeira. Eles eliminam a parte mecânica da avaliação. Entretanto, ter uma visão clara do que está acontecendo com o processo de assimilação dos assuntos vistos em classe,

exige uma visão mais profunda do desempenho dos alunos (VALENTE, 1999).

### 2.2.3 Simulações

Simulações são representações de objetos reais de um sistema ou evento. São modelos simbólicos e representativos da realidade que devem ser utilizados a partir da caracterização dos aspectos essenciais do fenômeno. Isto significa que a simulação deve ser utilizada após a aprendizagem de conceitos e princípios básicos do tema em questão.

A simulação induz a um nível intermediário entre o abstrato e o concreto, oferecendo a possibilidade do aluno desenvolver hipóteses, testá-las, analisar resultados e refinar os conceitos. Esta modalidade de uso do computador na educação é muito útil para trabalhos em grupo, principalmente os programas que envolvem decisões (VALENTE, 1999).

Com o uso desses softwares, o professor poderá promover ambientes de intensa interatividade, motivação e produtividade ao mesmo tempo que avalia o processo de ensino-aprendizagem. Essa forma de trabalho reverte o sentido da avaliação, que geralmente se resume a análise dos resultados e não do processo. Este último sendo o momento mais rico para as intervenções pedagógicas (OLIVEIRA; COSTA; MOREIRA, 2001).

Esses softwares têm como característica o controle das seqüências reprodutoras do evento pelo aluno, facilitando a simulação da realidade; o uso de ilustrações, de cor, animação e recursos sonoros para fornecer dados mais reais ao aluno, suprindo deficiências que a palavra escrita possa apresentar; fornece os resultados ao aluno tanto parcialmente quanto ao final da simulação, dentre outras.

### 2.2.4 Jogos

Os jogos têm como característica a existência de recursos motivacionais para despertar, manter e fixar a atenção do aluno.

Esses softwares devem ser fonte de recreação com vista à aquisição de um determinado tipo de aprendizagem, envolvendo elementos de desafio ou competição. Possuem uma grande capacidade de capturar a atenção do aluno no decorrer da tarefa, devido ao seu aspecto colorido, dinâmico e divertido. Além disso, oferecem aos usuários intensa interatividade permitindo ampliar relações sociais no ambiente de ensino, cativando o interesse dos alunos em relação a temas muitas vezes difíceis de serem apresentados por outra abordagem (OLIVEIRA; COSTA; MOREIRA, 2001). No entanto para VALENTE (1999), o grande problema com os jogos é que

estes por sua vez, por se tratarem de uma competição, pode desviar a atenção da criança do conceito envolvido no jogo.

Já OLIVEIRA, COSTA e MOREIRA (2001), acreditam que os jogos tem como objetivos, não apenas possibilitar entretenimento para o usuário, mas também influenciar o desenvolvimento socioafetivo e cognitivo do indivíduo, porque tem como essência a aprendizagem com prazer e a criatividade com diversão.

### 2.2.5 Aplicativos

São programas voltados para aplicações específicas, como processadores de texto, planilhas eletrônicas, e gerenciadores de banco de dados. Embora não tenham sido desenvolvidos para uso educacional, permitem interessantes usos em diferentes ramo do conhecimento.

VALENTE (1999) defende que, nos processadores de textos, as ações do aprendiz podem ser analisadas em termos do ciclo descrição - execução - reflexão - depuração - descrição. Quando o aprendiz está digitando um texto no processador de texto, a interação com o computador é mediada pelo idioma materno e pelos comandos de formatação. Apesar de simples de serem usados e de facilitar a expressão do pensamento, o processador de texto não pode executar o conteúdo do mesmo e apresentar um feedback do conteúdo e do seu significado para o aprendiz. A única possibilidade, em se tratando de reflexão, é comparar as idéias originais do formato com o resultado apresentado, não dando margem para a reflexão e depuração do conteúdo. Nesse sentido, o processador de textos não dispõe de características que auxiliam o processo de construção do conhecimento e a compreensão das idéias.

### 2.2.6 Multimídia e Internet

Em relação à multimídia, VALENTE (1999) chama a atenção para a diferenciação entre o uso de uma multimídia já pronta e o uso de sistemas de autoria para o aprendiz desenvolver sua multimídia.

Na primeira situação, o uso de multimídia é semelhante ao tutorial, apesar de oferecer muitas possibilidades de combinações com textos, imagens e sons. A ação do aprendiz se resume em escolher opções oferecidas pelo software. Após a escolha, o computador apresenta a informação disponível e o aprendiz pode refletir sobre a mesma. Às vezes o software pode oferecer também ao aprendiz, oportunidade de selecionar outras opções e navegar entre elas. Essa idéia pode manter o aprendiz ocupado por um certo tempo e não oferecer-lhe oportunidade de compreender e aplicar de modo significativo as informações selecionadas.

Dessa forma, o uso de multimídia pronta e Internet são atividades que auxiliam o aprendiz a adquirir informações, mas não a compreender ou construir conhecimentos com a informação obtida. Torna-se necessária a intervenção do "agente de aprendizagem" para que o conhecimento seja construído.

Na segunda situação, os programas são construídos para permitir o fácil desenvolvimento de tutoriais, viabilizando que professores não especializados em informática possam desenvolver SE de qualidade (OLIVEIRA; COSTA; MOREIRA, 2001). Integram de forma fácil, texto, imagem e som por meio de uma linguagem computacional de manipulação de ícones, links de hipertexto e telas gráficas.

No mercado brasileiro estão disponíveis o Hyperstudio, o Flash e o Visual Class, os quais vêm sendo utilizados pelos especialistas em informática na educação (OLIVEIRA; COSTA; MOREIRA, 2001).

### 2.2.7 Linguagens de Programação

Esses softwares permitem que pessoas, professores ou alunos, criem seus próprios protótipos de programas, sem que tenham que possuir conhecimentos avançados de programação.

Existem vários sistemas comerciais que incorporam diferentes tipos de linguagem, no entanto, no contexto educativo essas linguagens se prestam ao desenvolvimento de SE, mas a sua utilização não prescinde de profissionais da área de computação (OLIVEIRA; COSTA; MOREIRA, 2001).

O Logo (PAPERT, 1985), é um exemplo de linguagem de programação, em que as suas características disponíveis no seu processo de programação ajudam o aprendiz a encontrar seus erros, e ao professor compreender o processo pelo qual o aprendiz construiu conceitos e estratégias envolvidas no programa. O Logo propõe um ambiente de aprendizagem no qual o conhecimento não é meramente passado para o aluno, mas, uma forma de trabalho onde esse aluno em interação com os objetos desse ambiente, possa desenvolver outros conhecimentos, como por exemplo: conceitos geométricos ou matemáticos. Além disso, propicia ao aluno a possibilidade de aprender fazendo, ou seja, ensinando a tartaruga a resolver um problema, seguindo a linguagem de programação.

Levando em consideração os diversos tipos de softwares educativos apresentados, abordaremos a seguir alguns tipos de metodologias existentes na literatura que auxiliam o indivíduo (professor), na tarefa de avaliar.

## 2.3 Tipos de Metodologias Aplicadas para Avaliação de Software

Nesta seção, iremos abordar alguns dos tipos de metodologias existentes para avaliações de softwares. Cada metodologia foca distintos fatores, mas que ao final possuem um único objetivo, identificar um software que seja capaz de atender as reais necessidades do usuário.

Iremos agora, detalhar cada método que resolvemos abordar neste trabalho, são eles: Reeves, Ticese, Ergolist, Maep e Softmat.

### 2.3.1 Método de Reeves

A metodologia proposta por REEVES e HARMON (1996) apresenta duas abordagens complementares na avaliação de software educativo. Uma delas se baseia em quatorze critérios e a outra em dez critérios relacionados à interface com usuário. Os critérios são avaliados através de uma marca sobre uma escala não dimensionada representada por uma seta dupla. Em cada extremidade da seta são colocados os conceitos antagônicos que caracterizam o critério, de modo, que na extremidade esquerda fica situado o conceito mais negativo. A conclusão a respeito da avaliação é obtida graficamente analisando a disposição dos pontos marcados nas setas que devem ser ligados, colocando-se as setas umas sobre as outras.

### 2.3.2 Método Ticese

A Técnica de Inspeção Ergonômica de Software Educacional (TICESE) segundo GAMEZ (1998), é uma técnica que está em desenvolvimento no Laboratório de Utilizabilidade (LABIUTIL) da UFSC, e destina-se a apoiar os processo de avaliação do software educacional. A técnica favorece a elaboração de um laudo técnico com o objetivo de orientar os responsáveis, na instituição de ensino, sobre a aquisição de software, para o uso em contexto escolar. Nesta perspectiva, pretendeu-se criar uma técnica para a avaliação da conformidade ergonômica.

A técnica é formada por um conjunto específico de critérios de análise e tem seu suporte teórico nas ciências cognitivas, ergonomia de software, psicologia da aprendizagem e pedagogia. Aos critérios, está associado um conjunto de questões que visa orientar o(s) avaliador(s) na difícil tarefa de inspecionar as qualidades ergonômico/pedagógicas do software educacional.

Três módulos compõem a técnica. O módulo de classificação, de avaliação e de contextualização.

**Classificação**

É introdutório. Tem como objetivo determinar a modalidade de software educativo (Tutorial, Exercício e prática, simulador, hipertexto, ou outra classificação), a identificação da abordagem pedagógica subjacente, (Construtivista, Behaviorista, Construcionista, ou outra) e por fim, a identificação das habilidades cognitivas exigidas (aplicação, análise, síntese, e avaliação extensiva da Taxonomia de Bloom).

**Avaliação**

Consiste no principal módulo da técnica: avalia a conformidade do software educativo aos padrões ergonômicos de qualidade objetivando, assim, avaliar a capacidade do software em auxiliar o aprendizado específico. Através deste módulo é possível verificar os recursos pedagógicos e de apoio à aprendizagem utilizados e sua forma de operação. Neste caso, o módulo apóia também a avaliação da facilidade de uso do sistema e dos materiais impressos que o acompanham.

Os critérios definidos para efetuar esta inspeção foram desenvolvidos a partir de uma abordagem de convergência e de extensão dos critérios ergonômicos para interface de software em geral, propostos por BASTIEN e SCAPIN (1993).

As atividades segundo esta estratégia proposta por BASTIEN e SCAPIN (1993), basearam-se em uma revisão bibliográfica aprofundada sobre o tema e na aplicação dos critérios em desenvolvimento em situações de avaliação real de software educativo. O conjunto de critérios resultante é:

**i)** Qualidade da apresentação da informação

- abrangência dos dados de identificação (do produto, dos objetivos e pré-requisitos técnicos e pedagógicos);
- organização e apresentação da documentação impressa (presteza, agrupamento de itens, legibilidade e densidade informacional);
- organização e apresentação da Informação on line (presteza, legibilidade, agrupamento/distinção de itens, feedback imediato);
- significado dos códigos e denominações;
- homogeneidade/coerência.

**ii)** Qualidade dos recursos

- qualidade dos recursos para a motivação e compreensão dos conteúdos;
- qualidade dos Recursos de avaliação do aprendizado;
- qualidade da gestão de erros (correção, qualidade das mensagens e proteção contra os erros);
- qualidade da ajuda on-line.

**iii)** Qualidade da operação

- carga de trabalho (carga e densidade informacional, objetividade, ações mínimas);
- adaptabilidade (flexibilidade, consideração da experiência do usuário);
- controle explícito (ações explícitas e controle do usuário);
- compatibilidade.

**Contextualização**

É complementar ao critério anterior e visa auxiliar no processo de tomada de decisão sobre uma provável aquisição, mediante a adequabilidade do produto ao contexto específico da instituição. Cada instituição de ensino possui características e contextos próprios, que se diferenciam das demais. Apresentam projetos políticos pedagógicos próprios e, em geral, os recursos financeiros variam conforme suas disponibilidades.

A decisão sobre a aquisição do software não pode ser baseada unicamente na qualidade do produto em si, mas estar fundamentada sob uma série de considerações que avaliam a pertinência e adequabilidade do uso do software educacional na referida instituição, como enfatiza SILVA (1998).

### 2.3.3 Método Ergolist

O Ergolist é um sistema de avaliação de qualidade ergonômica de software, que foi desenvolvido pelo Laboratório de Utilizabilidade da Universidade Federal de Santa Catarina (LABIUTIL), para ser usado na internet, de forma on-line, segundo SILVA e VARGAS (1999).

O Ergolist se constitui em um serviço Web composto de uma base de conhecimento em ergonomia, associada a um checklist para a inspeção ergonômica de interfaces homem-computador.

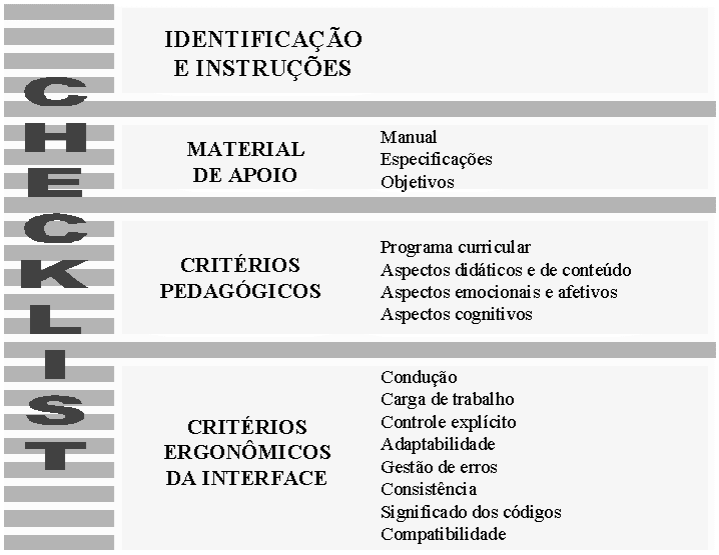


Figura 1: Estrutura do *Checklist.* (SILVA, 2002)

Nesse ambiente, o analista pode avaliar a interface de um aplicativo utilizando o checklist disponibilizado em um endereço Web administrado pelo LabIUtil.

Para HEEMANN (1997), essa metodologia baseia-se em recomendações ergonômicas geradas a partir de critérios tais como: importância e pertinência da recomendação, existência de material explicativo sobre ela e sua aderência a um critério ergonômico e a uma característica de interface. A sua utilização ocorre da seguinte maneira: O usuário (avaliador) dispõe de um checklist, onde poderá fornecer suas opiniões e observações à medida que for realizando a avaliação. Ao finalizar a avaliação, será fornecido um laudo pelo programa com os resultados estatísticos da avaliação realizada.

### 2.3.4 Método MAEP

O MAEP é um método de avaliação ergopedagógico proposto em SILVA (2002), no qual foi elaborado um modelo de avaliação que agrega aspectos pedagógicos e ergonômicos na mesma ferramenta, visando contemplar um projeto educacional, comunicacional e computacional que atendesse às exigências de um produto educacional informatizado e que pudesse ser estruturado de modo a auxiliar tanto projetistas como educadores na concepção, avaliação e utilização. A Figura 1 mostra a estrutura desse modelo.

A estratégia inicial para construir esse modelo baseou-se no levantamento dos objetivos pedagógicos e ergonômicos, reunindo o maior número de informações significativas sobre essa interdisciplinaridade de forma que o avaliador pudesse ter uma visão geral dos elementos que deveria observar, facilitando-lhe a pesquisa sobre esses pontos e a aplicação das várias tipolo-

gias.

A partir desse modelo, desenvolveu-se um checklist básico para avaliação de um software educacional na área da construção civil (SILVA, 2002), composto de três (3) partes, teve por objetivo qualificar o programa a partir das características ergonômicas e pedagógicas desejáveis. Dessa forma, o percentual de cada tópico apresentado significa o grau de satisfação em relação a essas características.

Este método tem origem em constatações e indagações oriundas do cotidiano de uso da informática educativa, especialmente no uso do software educacional. Além disso, a construção desse método foi resultado de uma reflexão aprofundada visando conciliar as exigências do campo da ergonomia de IHC com a pedagogia.

### 2.3.5 Método Softmat

O Softmat é um repositório virtual de softwares educacionais apropriados para matemática do ensino médio, em que os seus software vem acompanhados de suas respectivas avaliações.

Baseado nesse repositório, uma equipe formada por professores de matemática, alunos de licenciatura em matemática, uma pedagoga, um aluno do curso superior de tecnologia em desenvolvimeto de software e uma doutora em engenharia de sistemas e computação, utilizou-se da metodologia proposta por GLADCHEFF, ZUFFI e SILVA (2001), que é específica para matemática do ensino fundamental, adaptando-a para o ensino médio através de diversas modificações, originando assim a metodologia SoftMat.

Trata-se de um instrumento de avaliação de softwares educacionais voltados para matemática do ensino médio. É composto de um questionário, disposto em 5 blocos de questões, considerando tanto aspectos técnicos das normas ISO (ISO/IEC9126-1 e ISO/IEC12119) quanto questões específicas do setor educacional. Através deste instrumento são avaliados atributos de qualidade externa dos softwares.

Os blocos estão organizados da seguinte forma:

**A:** questões relativas à documentação (documentação de descrição e manual do usuário, impresso ou on line);

**B:** questões operacionais (relacionadas à instalação e utilização do software);

**C:** questões relacionadas a características pedagógicas gerais (objetivos, usabilidade, conteúdos matemáticos e praticidade);

**D:** questões relacionadas às propostas dos Parâmetros Curriculares do Ensino Médio (PCNEM) para Matemática;

**E:** questões relativas à proposta pedagógica privilegiada no software.

Na visão de BATISTA (2004), a metodologia SoftMat apresenta, ainda, uma questão aberta, que encerra a avaliação solicitando o registro da visão do avaliador a respeito do software avaliado (pontos considerados positivos e negativos, importância do software como recurso didático, entre outros).

### 2.3.6 Discussão

As metodologias apresentadas, bem como os debates sobre a avaliação de software educativo, assunto discutido neste trabalho, demonstram quão ampla e complexa é a tarefa de se estabelecer parâmetros gerais de avaliação nos seus diferentes aspectos, reforçando a necessidade de aprofundar e criar mecanismos integradores que facilitem esse processo e o torne menos custoso para avaliadores não especializados nesse domínio.

Dessa forma, apresentamos uma síntese de cada metodologia, abordando o que elas apresentam em comum.

O método de Reeves, é um método que utiliza duas abordagens sendo uma baseada em quatorze critérios e a outra em dez critérios, mas ambos relacionados à interface com o usuário. Quanto aos métodos Ticese e Ergolist, ambos foram desenvolvidos pela UFSC, sendo que o primeiro tem por finalidade o desenvolvimento de um laudo técnico, para orientar o avaliador na difícil tarefa de inspecionar as qualidades ergonômicas/pedagógicas. Já o segundo, tem por finalidade a criação de um *checklist* para ser utilizado de forma *on-line*, como forma de avaliar a qualidade ergonômica do software.

O método MAEP é um método desenvolvido para avaliar aspectos ergonômicos e pedagógicos do software, no intuito de se realizar uma análise minuciosa não somente de sua avaliação mas também de sua criação.

Por fim, o método Softmat é um repositório virtual específico para matemática do ensino médio, onde considera tanto aspectos técnicos quanto específicos do setor educacional, como é o caso dos parâmetros curriculares do ensino médio (PCNEM). Este por sua vez, defende que os educandos desenvolvam a capacidade de analisar e julgar a resolução de problemas. Esta capacidade corresponde a uma visão de aprendizagem denominada “construção de competências”, que também é defendida por Vergnaud em sua Teoria, apresentada no capítulo a seguir.

# 3 Teoria dos Campos Conceituais

Neste capítulo, descreveremos a Teoria dos Campos Conceituais de Vergnaud e suas implicações para o ensino, bem como a sua influência neste trabalho.

## 3.1 Introdução

A Teoria dos Campos Conceituais é uma teoria psicológica cognitivista que supõe que o núcleo do desenvolvimento cognitivo é a conceitualização do real. Essa teoria foi desenvolvida por Gerard Verganud com o intuito de ampliar e redirecionar o foco piagetiano das operações lógicas gerais, bem como as estruturas gerais do pensamento, com o intuito de estudar o funcionamento cognitivo do sujeito-em-situação (MOREIRA, 2002).

Vergnaud era discípulo de Piaget, e toma como premissa que o conhecimento está organizado em campos conceituais cujo domínio, por parte do sujeito, ocorre ao longo de um largo período de tempo, através de experiência, maturidade e aprendizagem. Embora Vergnaud esteja especialmente interessado nos campos conceituais das estruturas aditivas e das estruturas multiplicativas (VERGNAUD, 1983), a Teoria dos Campos Conceituais não é específica desses campos, nem da Matemática.

Essa teoria não é, no entanto, uma teoria de ensino de conceitos explícitos e formalizados. Trata-se de uma teoria psicológica do processo de conceitualização do real que permite localizar e estudar continuidades e rupturas entre conhecimentos do ponto de vista de seu conteúdo conceitual (MOREIRA, 2002). No estudo desse processo, qualquer reducionismo é perigoso na medida em que a conceitualização do real é específica de conteúdo e não pode ser reduzida nem às operações lógicas gerais, nem às operações puramente lingüísticas, nem à reprodução social, nem à emergência de estruturas inatas, nem, enfim, ao modelo do processamento da informação.

A Teoria dos Campos Conceituais, por se tratar de uma teoria bastante importante dentro do âmbito educacional, é um dos pilares deste trabalho, juntamente com técnicas extraídas da

engenharia de software. Fazemos uso de seus conceitos a fim de traçamos algumas correlações que sirvam de alicerces na escolha de um software educativo adequado. No capítulo 5 deste trabalho, apresentamos uma tabela, denominada Tabela de Requisitos de Dominio, que irá relacionar a tríade (S,I,R) citada por Vergnaud, com a nossa proposta de metodologia (correlações), no intuito de facilitar essa tarefa de se avaliar software educativo, que possuam a mesma área de aplicação.

A seguir, apresentaremos os conceitos-chave da Teoria dos Campos Conceituais os quais são, os conceitos de esquema (a grande herança piagetiana de Vergnaud), situação, invariante operatório (teorema-em-ação ou conceito-em-ação), e a sua concepção de conceito.

## 3.2 Conceitos

Vergnaud define conceito como um tripleto C = (S, I, R) onde:

S é um conjunto de situações que dão sentido ao conceito; I é um conjunto de invariantes (objetos, propriedades e relações) sobre os quais repousa a operacionalidade do conceito, ou o conjunto de invariantes operatórios associados ao conceito, ou o conjunto de invariantes que podem ser reconhecidos e usados pelos sujeitos para analisar e dominar as situações do primeiro conjunto; R é um conjunto de representações simbólicas (linguagem natural, gráficos e diagramas, sentenças formais etc.) que podem ser usadas para indicar e representar esses invariantes e, conseqüentemente, representar as situações e os procedimentos para lidar com elas.

Uma definição pragmática poderia considerar um conceito como um conjunto de invariantes utilizáveis na ação, mas esta definição implica também um conjunto de situações que constituem o referente e um conjunto de esquemas postos em ação pelos sujeitos nessas situações. Daí, o tripleto (S, R, I) onde, em termos psicológicos, S é a realidade e (I, R) a representação que pode ser considerada como dois aspectos interagentes do pensamento, ou seja, o significado, que é o conjunto de invariantes que constituem as propriedades do conceito e o significante, que é o conjunto de formas simbólicas ou lingüísticas que permitem a representação do conceito (VERGNAUD, 1983).

Isso implica que para estudar o desenvolvimento e uso de um conceito, ao longo da aprendizagem ou de sua utilização, é necessário considerar esses três conjuntos simultaneamente. Por tudo isso, é necessário falar-se em campos conceituais. Mas se os conceitos tornam-se significativos através de situações decorre, naturalmente, que as situações e não os conceitos constituem a principal entrada de um campo conceitual.

Um campo conceitual é, em primeiro lugar, um conjunto de situações (MOREIRA, 2002), cujo domínio requer o conhecimento de vários conceitos de naturezas distintas.

## 3.3 Situações

O conceito de situação ao qual Vergnaud refere-se, não é o de situação didática, mas sim o de tarefa, sendo que toda situação complexa pode ser analisada como uma combinação de tarefas, para as quais é importante conhecer suas naturezas e dificuldades próprias. A dificuldade de uma tarefa não é nem a soma nem o produto das diferentes subtarefas envolvidas, mas é claro que o desempenho em cada subtarefa afeta o desempenho global.

Sendo assim, as situações é que dão sentido ao conceito; as situações é que são responsáveis pelo sentido atribuído ao conceito (VERGNAUD, 1990), mas o sentido não está nas situações em si mesmas, assim como não está nas palavras nem nos símbolos. O sentido é uma relação do sujeito com as situações e com os significantes. Mais precisamente, são os esquemas, ou seja, os comportamentos e sua organização, evocados no sujeito por uma situação ou por um significante (representação simbólica) que constituem o sentido dessa situação ou desse significante para esse indivíduo. Por exemplo, o sentido de adição para um sujeito individual é o conjunto de esquemas que ele pode utilizar para lidar com situações com as quais se defronta e que implicam a idéia de adição; é também o conjunto de esquemas que ele pode acionar para operar sobre os símbolos numéricos, algébricos, gráficos e lingüísticos que representam a adição.

Dessa forma, podemos afirmar que a idéia de campo conceitual nos levou ao conceito de conceito como um tripleto (referente, significado e significante); porém, como são as situações que dão sentido ao conceito, chegamos ao conceito de situação e dele ao de esquema, pois são os esquemas evocados no sujeito que dão sentido a uma dada situação. O conceito de esquema, como veremos, nos levará ao conceito de invariante operatório.

## 3.4 Esquemas

Para Vergnaud, esquema é a organização invariante do comportamento para uma determinada classe de situações onde se devem pesquisar os conhecimentos-em-ação do sujeito, isto é, os elementos cognitivos que fazem com que a ação do sujeito seja operatória. Esquema é o conceito introduzido por Piaget para dar conta das formas de organização tanto das habilidades sensório-motoras como das habilidades intelectuais. Um esquema gera ações e deve conter regras, mas não é um estereótipo porque a seqüência de ações depende dos parâmetros

da situação (VERGNAUD, 1983). Um esquema é um universal que é eficiente para toda uma gama de situações e pode gerar diferentes seqüências de ação, de coleta de informações e de controle, dependendo das características de cada situação particular. Não é o comportamento que é invariante, mas a organização do comportamento. Há esquemas perceptivo-gestuais como o de contar objetos, ou de fazer um gráfico ou um diagrama, mas há também esquemas verbais, como o de fazer um discurso, e esquemas sociais, como o de seduzir outra pessoa ou o de gerenciar um conflito. Algoritmos, por exemplo, são esquemas, mas nem todos os esquemas são algoritmos.

Vergnaud considera que os esquemas necessariamente se referem a situações, a tal ponto que, segundo ele (VERGNAUD, 1990), dever-se-ia falar em interação esquema-situação ao invés de interação sujeito-objeto da qual falava Piaget.

Para Vergnaud, os esquemas referem-se necessariamente a situações, ou classes de situações, onde ele (VERGNAUD, 1990) distingue entre:

1. classes de situações em que o sujeito dispõe, no seu repertório, em dado momento de seu desenvolvimento e sob certas circunstâncias, das competências necessárias ao tratamento relativamente imediato da situação;

2. classes de situações em que o sujeito não dispõe de todas as competências necessárias, o que obriga a um tempo de reflexão e exploração, a hesitações, a tentativas frustradas, levando-o eventualmente ao sucesso ou ao fracasso.

O conceito de esquema não funciona do mesmo modo nas duas classes. Na primeira delas, observam-se, para uma mesma classe de situações, condutas amplamente automatizadas, organizadas por um só esquema, enquanto que na segunda observa-se a sucessiva utilização de vários esquemas, que podem entrar em competição e que, para atingir a meta desejada, devem ser acomodados, descombinados e recombinados.

De um modo geral, todas as condutas comportam uma parte automatizada e uma parte de decisão consciente. Os esquemas são freqüentemente eficazes, mas nem sempre efetivos. Quando o sujeito usa um esquema ineficaz para uma certa situação, a experiência o leva a mudar de esquema ou a modificar o esquema. Está aí a idéia piagetiana de que os esquemas estão no centro do processo de adaptação das estruturas cognitivas, ou seja, na assimilação e na acomodação. Contudo, Vergnaud dá ao conceito de esquema um alcance muito maior do que Piaget e insiste em que os esquemas devem relacionar-se com as características das situações às quais se aplicam.

Segundo Vergnaud, as expressões conceito-em-ação e teorema-em-ação designam os conhecimentos contidos nos esquemas, de forma a serem designados pela expressão mais global invariantes operatórios. Teorema-em-ação é uma proposição considerada como verdadeira sobre o real; conceito-em-ação é uma categoria de pensamento considerada como pertinente. Dessa forma, para uma criança de cinco anos enumerar uma pequena coleção de objetos, por mais que varie a forma de contar, por exemplo, copos na mesa, cadeiras da sala, pessoas sentadas de maneira esparsa em um jardim, não deixa de haver uma organização invariante para o funcionamento do esquema: coordenação dos movimentos dos olhos e gestos dos dedos e das mãos, enunciação correta da série numérica, identificação do último elemento da série como o cardinal do conjunto enumerado (acentuação ou repetição do último "número" pronunciado).

Naturalmente, os esquemas usados por crianças maiores e por adultos em determinadas classes de situações podem ser muito mais elaborados, mas a idéia é mesma, ou seja o esquema é a forma estrutural da atividade, é a organização invariante do sujeito sobre uma classe de situações dadas e contém conhecimentos-em-ação que são implícitos.

## 3.5 Invariantes Operatórios

Designam-se pelas expressões "conceito-em-ação" e "teorema-em-ação" os conhecimentos contidos nos esquemas. Pode-se também designá-los pela expressão mais abrangente "invariantes operatórios" (VERGNAUD, 1983).

Esquema é a organização da conduta para uma certa classe de situações; teoremas-em-ação e conceitos-em-ação são invariantes operacionais, logo, são componentes essenciais dos esquemas e determinam as diferenças entre eles.

Teorema-em-ação é uma proposição tida como verdadeira sobre o real. Conceito-em-ação é um objeto, um predicado, ou uma categoria de pensamento tida como pertinente, relevante. Em geral, os alunos não são capazes de explicar ou mesmo expressar em linguagem natural seus teoremas e conceitos-em-ação.

Na abordagem de uma situação, os dados a serem trabalhados e a seqüência de cálculos a serem feitos dependem de teoremas-em-ação e da identificação de diferentes tipos de elementos pertinentes. A maioria desses conceitos e teoremas-em-ação permanecem totalmente implícitos, mas eles podem também ser explícitos ou tornarem-se explícitos e aí entra o ensino: ajudar o aluno a construir conceitos e teoremas explícitos, e cientificamente aceitos, a partir do conhecimento implícito.

É nesse sentido que conceitos-em-ação e teoremas-em-ação podem, progressivamente, tornarem-se verdadeiros conceitos e teoremas científicos, mas isso requer tempo.

## 3.6 Discussão

A avaliação de SE deve considerar com mais ênfase e de forma bem fundamentada a relação entre o uso do software e a aprendizagem de conceitos (GOMES et al., 2002). No que concerne à aprendizagem da matemática, os softwares mais proveitosos seriam aqueles que permitem uma grande interação do aluno com os conceitos ou idéias matemáticas, propiciando a descoberta, inferindo resultados, levantando e testando hipóteses, além de criar soluções-problemas.

Em geral, pesquisadores e professores têm dificuldade em entender que a compreensão de um conceito, por mais simples que seja, não emerge apenas de um tipo de situação, assim como uma simples situação sempre envolve mais que um único conceito (MAGINA et al., 2001).

Dessa forma, considerando a Teoria dos Campos Conceituais, nota-se que a maioria dos softwares destinados à educação matemática parece evocar apenas uma estreita porção de um campo conceitual específico, sendo relevante facilitar a emergência de um grande número de situações que darão significado aos conceitos matemáticos (MAGINA et al., 2001).

Ela considera que existe uma série de fatores que influenciam e interferem na formação e desenvolvimento dos conceitos e que o conhecimento conceitual deve emergir dentro de situações-problemas. Em outras palavras, nem um só conceito nem uma situação isolada dá conta do processo de aquisição de um conhecimento (GOMES et al., 2002).

Portanto, um dos principais problemas do ensino de matemática é introduzir na sala de aula uma melhor relação entre conceitos e a resolução de problemas, de maneira a torná-los interessantes e compreensíveis para os alunos. E isto só é possível, se o professor conseguir fazer com que seus alunos desenvolvam duas ferramentas essenciais que Vergnaud denomina de competência e concepção, onde a primeira diz repeito a ação do aluno diante das situações e a segunda diz respeito as expressões verbais ou expressões simbólicas traçadas pelo aluno.

Por isso resolvemos adotar a Teoria dos Campos Conceituais como um dos alicerces desse trabalho, pois ela defende que todos os fatores que interferem no sucesso do aprendizado do aluno devem ser analisados. E para consolidar o uso dessa teoria na metodologia que estamos propondo, resolvemos associá-la a engenharia de requisitos que apresentamos no capítulo a seguir e que juntas formam a base desse trabalho.

# 4 Engenharia de Requisitos

Antes de apresentarmos a metodologia que propusemos é importante deixar claro quais as funções dos requisitos no processo de desenvolvimento de um software sob o ponto de vista preciso da Engenharia de Requisitos. Além disso, ressaltaremos aqui a abordagem que utiliza pontos de casos de uso, proposta por KARNER (1993) para a realização de estimativas de tempo e esforço de desenvolvimento de produtos de software, baseado nas informações dos requisitos.

## 4.1 Requisitos de Software

Fundamental para compreendermos a Engenharia de Requisitos é entendermos exatamente o que são requisitos de um sistema. Sendo assim, podemos descrever requisitos como as funções que deverão ser incorporadas pelo software, quando inserido em seu contexto de funcionamento. No entanto, necessidades como desempenho, integridade, disponibilidade e segurança seriam difíceis de acomodar nessa definição preliminar de requisitos. Apesar de tais qualidades transformarem-se, em algum nível de abstração, em funcionalidades do novo sistema, no nível de abstração no qual requisitos estão sendo tratados, tais qualidades não são funcionalidades do sistema em especificação. Dessa forma, segundo SOMMERVILLE (2003), podemos classificar os requisitos de um software da seguinte forma:

1. Requisitos Funcionais: são declarações de funções que o sistema deve fornecer, como o sistema deve agir a entradas específicas e como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também explicitamente declarar o que o sistema não deve fazer;

2. Requisitos Não funcionais: são restrições sobre os serviços ou as funções oferecidos pelo sistema. Entre eles, destacam-se restrições sobre o processo de desenvolvimento, padrões, entre outros. Na Figura 2 é definida uma classificação para os requisitos não funcionais;

3. Requisitos de Domínio: são requisitos que definem funções específicas de determinados domínios de aplicação. Esses requisitos tanto podem ser funcionais como não funcionais.

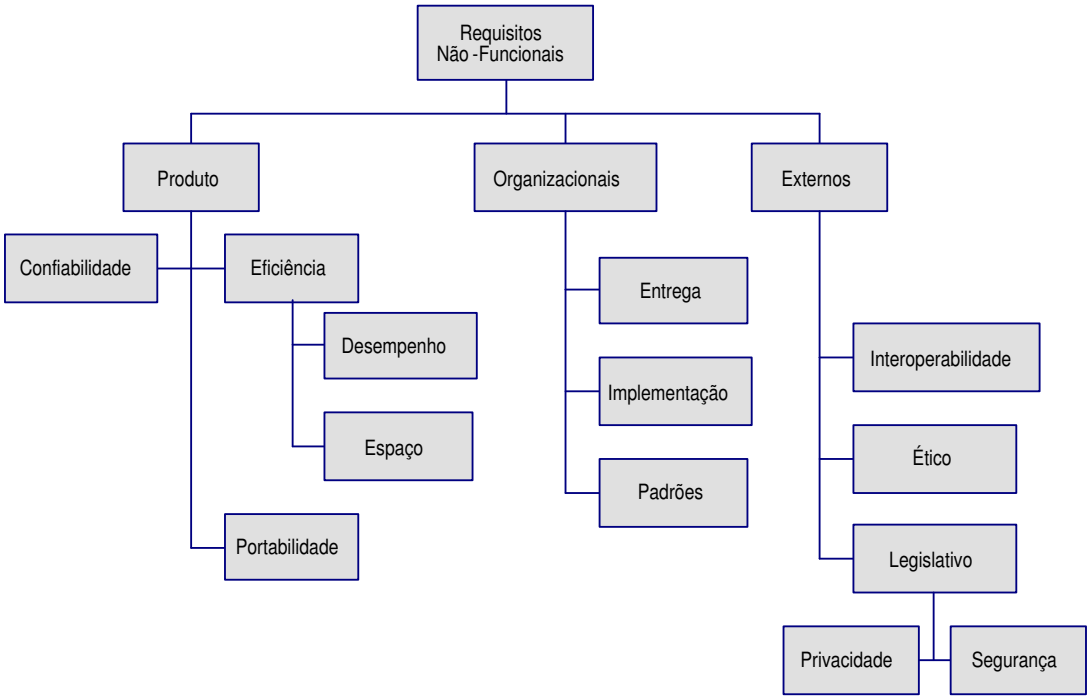


Figura 2: Classificação de Requisitos Não Funcionais.

Normalmente, os requisitos funcionais são verificados em um determinado software apenas pela sua presença ou ausência. Já os requisitos não funcionais, não podem ser avaliados da mesma forma que os funcionais. De fato, a verificação de requisitos não funcionais não é uma tarefa simples, pois muitas vezes esses requisitos são também tratados como funções a serem fornecidos pelo sistema, sendo essas funções difíceis de serem testadas nos produtos de software. Para resolver esses problemas, os requisitos não funcionais devem ser expressos quantitativamente segundo SOMMERVILLE (2003), utilizando métricas que possam ser efetivamente testadas.

## 4.2 Casos de Uso

Um caso de uso descreve as operações que o sistema deve cumprir para cada usuário. Ele vai ajudar a formalizar as funções que o sistema precisa realizar. Assim, deve haver um caso de uso para cada tarefa que o sistema deve cumprir para um usuário.

Por exemplo, para um sistema de reservas, podem haver casos de uso para fazer uma reserva, consultar as reservas, suprimir uma reserva, imprimir relatórios etc.

Um caso de uso se apresenta como uma lista completa das interações entre um usuário e o sistema para cumprir uma tarefa. O caso de uso a seguir mostra todas as iterações de um usuário para a realização de um saque em um caixa eletrônico.

1. O usuário introduz o cartão no caixa eletrônico
2. O caixa eletrônico propõe varias operações
3. O usuário aperta o botão “‘saque”
4. O usuário escolhe a conta (ex.: “conta corrente”)
5. O usuário entra a valor do saque
6. O usuário entra sua senha
7. O caixa eletrônico verifica a senha com o banco e o saldo da conta caixa eletrônico da o dinheiro para o usuário
8. O caixa eletrônico imprime um recibo

Casos de uso descrevem interações entre o sistema e atores. Um ator é “alguma coisa” (usuário, outro sistema, ...) que não faz parte do sistema e interage com este. Por exemplo, para um sistema de regulagem da temperatura, o termostato pode ser um ator, visto que ele interage com o sistema e não faz parte dele.

Um usuário real pode cumprir vários papéis para o sistema, ou seja, ser representado por vários atores, por exemplo, num banco, o gerente pode ser um ator quando ele coloca dinheiro no caixa (ator: operador) e um outro ator quando ele retira dinheiro de sua própria conta (ator: cliente). Um ator pode também representar várias pessoas, quando falamos de um “cliente” retirando dinheiro do caixa eletrônico, é claro que esse ator representa qualquer pessoa.

É importante identificar todos os atores que vão interagir com o sistema. E para cada ator, é importante identificar que operações ele pode realizar. A definição de caso de usos de um sistema se faz a partir dos atores, e não a partir das operações. Isso quer dizer que não se deve procurar quaisquer funcionalidades abstratas que seria interessante para o sistema cumprir, ao invés disso deve-se procurar funcionalidades concretas que um usuário realmente precisa.

Geralmente, um caso de uso é iniciado por um ator, não pelo sistema. Por exemplo, no exemplo apresentado anteriormente, o primeiro passo é “O usuário introduz o cartão”’. Além disso, um caso de uso só descreve as interações entre o ator e o sistema, não descreve como essas funcionalidades vão ser implementadas. Assim, no exemplo anterior não está descrito como o sistema vai verificar a senha ou o saldo da conta. Isso será explicitado mais tarde, com outros diagramas.

Casos de uso têm vários objetivos importantes:

- Ser compreensíveis para usuários que provavelmente não entendem informática.
- Incentivar a análise do sistema especificando as funcionalidades necessárias.
- Delimitar o sistema.
- Servir de base para os casos de teste.

Casos de uso têm que ser compreensíveis por usuários pois só eles sabem o que o sistema realmente precisa fazer. Os casos de uso permitem verificar se o desenvolvedor e o usuário concordam sobre o que o sistema de fato deve fazer. Dessa forma, casos de uso podem servir de ”contratos” entre os usúarios e a equipe de desenvolvimento.

Casos de uso também podem ser usados como base para criar casos de teste. Testar o sistema para verificar se ele faz o que precisa e sem *bugs* é uma tarefa fundamental do desenvolvimento de software. Mas ao mesmo tempo, é difícil estabelecer bons testes. Os casos de uso são muito úteis nesse sentido, já que um caso de uso descreve uma interação completa e real com o sistema.

Os casos de uso são descrições muito abstrata das funcionalidades necessárias. Quando precisamos refinar mais a especificação do sistema, cada caso de uso poderá ser formalizado com diagramas de seqüência e/ou de colaboração. Esses diagramas são mais formais, e podem ser mais úteis para especificar precisamente (sem ambigüidades) o funcionamento do sistema. Mas, ao mesmo tempo, esses diagramas podem ser mais difíceis de entender pelo usuário. Normalmente, existem vários desses diagramas associados a um só caso de uso.

### 4.2.1 O Diagrama de Casos de Uso

O nome de um caso de uso pode ser qualquer sentença, mas UML (RUMBAUGH; JACOBSON; BOOCH, 1998) recomenda usar uma frase ativa curta (verbo + substantivo), por exemplo: “entrar reserva”, “retirar dinheiro” etc.

Esse diagrama é diferente dos outros porque contém poucos elementos gráficos. Um caso de uso é principalmente composto de um texto livre, ou pseudo-código que descreve cada interação. Os elementos principais do diagrama são uma elipse para representar um caso de uso e um pequeno boneco para representar um ator (Figura 3). O nome do caso de uso pode ser dentro da elipse ou abaixo dele.

O diagrama não especifica os casos de uso, mas só apresenta qual ator interage com qual caso e organiza os casos entre eles (ex.: qual herda de um outro). Os casos de uso vão ser

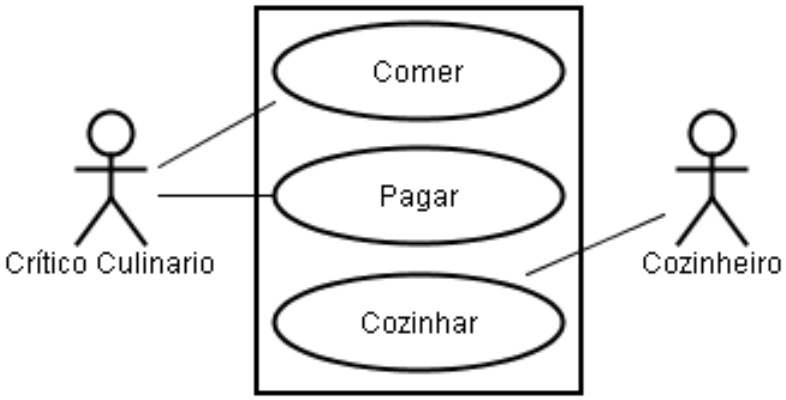


Figura 3: Exemplo de um Diagrama de Caso de Uso.

especificados em outros documentos. A UML não define precisamente a forma dessa descrição, contudo uma abordagem bastante utilizada é o uso de um texto livre organizado em seções, a seguir é apresentada uma possível estrutura para organização de uma especificação de caso de uso.

**Nome do Caso de Uso**

Descrição do caso de uso (um parágrafo).

**Atores**

Lista dos nomes dos atores com descrição curta.

**Prioridade**

Seja este caso de uso muito importante no projeto ou acessório?

**Estado**

Qual é o estado desse caso de uso (ainda muito abstrato, bem especificado, fechado, ...) ?

**Pré-Condições**

Lista de condições que têm que ser verificadas antes que o caso de uso começa.

**Fluxo de Eventos**

**Fluxo Principal**

1. Primeiro passo no caso de uso.

2. Segundo passo no caso de uso.

3. ...

**Fluxos Alternativos**

Descrever os fluxos alternativos.

**Pós-Condições**

Lista de condições que têm que ser verificadas depois do fim do caso de uso.

**Pontos de Extensão**

Lista dos pontos de extensão no caso de uso.

**Casos de Uso Incluídos**

Lista dos nomes dos casos de usos incluídos.

**Outros Requisitos**

Lista de outros requisitos que afetam o caso de uso.

### 4.2.2 Benefícios e Problemas dos Casos de Uso

Os casos de uso são um formato novo e mais ágil para capturar requerimentos de software. Eles contrastam com documentos maiores e "monolíticos" que tentam, mas falham completamente em conter todos os requerimentos possíveis antes do início da construção de um novo sistema.

Os casos de uso podem ser facilmente adicionados e removidos do projeto de software assim que as prioridades mudam, podendo também servir como base para estimar, escalonar e validar esforços. Uma razão porque os casos de uso se tornaram populares é que são fáceis de entender por pessoas da área de negócio, e assim provaram ser uma excelente ponte entre quem desenvolve o software e os utilizadores finais.

Os casos de uso também têm as suas dificuldades. São excelentes para capturar os requisitos funcionais de um sistema, mas não têm tanto sucesso em capturar requisitos não-funcionais.

## 4.3 Pontos de Casos de Uso

Atualmente, a análise de sistemas orientados a objetos utiliza diagramas de casos de uso para descrever as funcionalidades do sistema de acordo com a forma de utilização por parte dos usuários. A técnica de análise de dimensão por casos de uso foi criada para permitir que seja possível estimar o tamanho de um software ainda na fase de levantamento de requisitos,

utilizando-se dos próprios documentos gerados nesta fase de análise como subsídio para o cálculo dimensional. A técnica de estimativa por pontos de caso de uso foi proposta em 1993 por Gustav Karner (KARNER, 1993). Essa técnica trata de estimar o tamanho de um software de acordo com o modo como os usuários o utilizarão, a complexidade de ações requerida por cada tipo de usuário e uma análise em alto nível dos passos necessários para a realização de cada tarefa.

Os passos necessários para a geração da estimativa proposta por Karner, são:

1. Classificar os atores envolvidos em cada caso de uso, de forma a obter um somatório de pontos não-ajustado. O peso total dos atores do sistema (*Unadjusted Actor Weight*, ou UAW) é calculado pela soma dos produtos do número de atores de cada tipo pelo respectivo peso;

2. Calcular o peso bruto dos casos de uso (*Unadjusted Use Case Weight*, ou UUCW). O cálculo do UUCW é realizado como no cálculo de peso dos atores, somando-se os produtos da quantidade de casos de uso classificados em cada tipo pelo peso nominal do tipo em questão. O peso total não ajustado (*Unadjusted Use Case Points*, ou UUCP) é calculado pelo somatório entre os pesos de atores e casos de uso:

$$UUCP = UAW + UUCW$$

3. Calcular os fatores de ajuste. O método de ajuste é constituído de duas partes - um cálculo de fatores técnicos (*Technical Complexity Factor*, ou TCF), cobrindo uma série de requisitos funcionais do sistema; e um cálculo de fatores de ambiente (*Environment Factor*, ou EF), requisitos não-funcionais associados ao processo de desenvolvimento; Esses fatores de ajuste são calculados utilizando as seguintes fórmulas:

$$TCF = 0.6 + (.01 * TFactor)$$

$$EF = 1.4 + (-0.03 * EFactor)$$

4. Finalmente, podemos calcular o valor total do sistema em (*Use Case Points*, ou UCP) utilizando-se da seguinte fórmula:

$$UCP = UUCP * TFC * EF$$

É válido ressaltar que os pesos utilizados para a definição da complexidade de atores, de casos de uso e para a definição de fatores técnicos e ambientais são apresentados na literatura de forma mais ou menos padronizada. Esses pesos são definidos de forma empírica de acordo com e complexidade de um caso de uso ou da importância de um ator.

Na metodologia que estamos apresentando neste trabalho, utilizaremos valores aproximados aos pesos comumente utilizados para a definição desses elementos. Contudo, estamos condicionando os pesos à importância do fator que estamos avaliando e não à sua complexidade, visto que estamos utilizando a técnica de pontos de casos de uso para fins de avaliação e não para a realização de estimativa de esforço de desenvolvimento.

## 4.4 Outras Técnicas de Estimativa de Esforço

Apresentamos de maneira detalhada a Técnica de Pontos de Casos de Uso, que é uma técnica utilizada para estimar o esforço do desenvolvimento de um software. No entanto na literatura são apresentadas outras técnicas que também são utilizadas para estimar o esforço do desenvolvimento de um software. A seguir, apresentamos,de maneira sucinta, algumas dessas outras técnicas.

- Linhas de Código (LOC): A técnica de mensuração por linhas de código é uma das mais antigas medidas de tamanho de projeto de desenvolvimento de software. Ela consiste na contagem da quantidade do número de linhas de código de um programa de software. Além de ser muito simples é também muito fácil automatizar sua implementação, mas apresenta algumas desvantagens dentre as quais citamos: a dependência da linguagem, do software e do desenvolvedor (PRESSMAN, 2004); ausência de padrão de contagem e o fato de somente poder ser aplicada na fase de codificação.

- O Modelo SLIM (*Software LIfe Cycle Management*) (PUTNAM; MYERS, 1992): É um modelo de estimativa que busca medir esforço e prazo através da dinâmica de múltiplas variáveis que pressupõe distribuição de esforços específicos ao longo da existência de um projeto de software. Relaciona o número de linhas de código ao tempo e esforço de desenvolvimento. Uma desvantagem da técnica é sua vinculação a linguagem usada e a exigência de certo tempo para obter-se valores reais para os parâmetros da fórmula.

- Delphi: É uma técnica que se resume à consulta de especialistas de determinada área, em determinada linguagem e/ou determinado assunto para que, usando sua experiência e entendimento do projeto proposto, façam as estimativas devidas. Devem ser feitas várias

estimativas do mesmo projeto, pois é comum que elas carreguem influências e tendências dos especialistas. É um método empírico, baseado em experiências profissionais que podem ser subjetivas (H.A.LINSTONE; M.TUROFF, 1975).

- PSP (*Personal Software Process*) (HUMPHREY, 1991): É uma técnica derivada do SEI-CMM (*Software Engineering Institute Capability Matutiry Model*) que foi desenvolvida com a função de capacitar, melhorar e otimizar o processo individual de trabalho. A técnica divide-se em sete etapas, sendo que nas etapas PSP0, PSP0.1 e PSP1 estima-se o tamanho e o tempo necessário para o desenvolvimento do produto.

- Análise por Pontos de Função (ALBRECHT; GAFFNEY, 1983): Busca medir a complexidade do produto pela quantificação de funcionalidades expressa pela visão que o usuário tem do mesmo. O modelo mede o que é o sistema, o seu tamanho funcional e não como este será, além de medir a relação do sistema com usuários e outro sistemas. É independente da tecnologia usada e mede uma aplicação pelas funções desempenhadas para/e por solicitação do usuário final. Pontos de Função, na realidade, são o tamanho dos requisitos funcionais de um sistema.

- COCOMO (*COnstructive COst MOdel*) (BOEHM, 2000): Modelo desenvolvido para estimar o esforço de desenvolvimento, prazos e tamanho da equipe para projetos de software. Utiliza equações desenvolvidas por Boehm para prever o número de programadores-mês e o tempo de desenvolvimento; podem ser calculados usando medidas de Linhas de Código ou Pontos de Função. Devem ser realizados ajustes nas equações a fim de representar as influências sobre os atributos hardware e software durante o ciclo de vida do projeto.

As técnicas apresentadas acima são apenas algumas dentre as muitas existentes, sendo que cada uma abrange uma determinada área. Não existe uma métrica que completa o estudo por si só, desta forma, recomenda-se que seja utilizada a técnica mais adequada para medir projeto de software ou a utilização de mais de uma técnica em conjunto.

A técnica de Pontos de Casos de Uso (TPCU) é uma junção de duas técnicas, que é a Pontos de Função juntamente com a MK II (SYMONS, 1991). A técnica MK II é uma adaptação da técnica Ponto de Função, bastante utilizada na Inglaterra. Devido a essa junção, tornou-se possível fazer uma análise não apenas baseada em requisitos funcionais, como antes era proposto pela Técnica de Pontos de Função, mas agora poder analisar os requisitos não-funcionais dos mesmos. Levando isso em consideração, resolvemos fazer uso da TPCU como base para esse trabalho.

No capítulo a seguir, apresentamos como foi utilizado a Teoria dos Campos Conceituais juntamente com a Técnica de Pontos de Casos de Uso para dar origem a nossa metodologia de avaliação comparativa de SE.

# 5 Avaliação Comparativa de Software Educativo

O objetivo principal da metodologia de avaliação que estamos apresentando neste trabalho é permitir que um professor que esteja escolhendo quais produtos de software utilizar em sala de aula para um determinado domínio da matemática (campo conceitual), seja capaz de realizar a escolha de forma mais precisa. Por escolha mais precisa, queremos dizer que não somente aspectos da qualidade da interface gráfica, que em muitos casos acabam escondendo outros aspectos importantes com relação à usabilidade do software, possam ser decisivos no processo de escolha. Estamos propondo neste trabalho, uma forma mais aguçada de se aferir requisitos funcionais e não funcionais que sejam, a priori, relevantes para os produtos de software que estão sendo avaliados.

## 5.1 Metodologia de Avaliação Comparativa

Na Figura 4, a seguir, apresentamos de forma simplificada a metodologia de avaliação que estamos propondo. Nesse diagrama podemos observar as atividades envolvidas no processo de avaliação.

As seguintes são atividades da metodologia:

1. selecionar Software Candidato: nessa primeira atividade o software a ser avaliado é escolhido.
2. levantar Requisitos Funcionais: o levantamento dos requisitos funcionais deve ser realizado através da construção de diagramas de casos de uso considerando as funções/operações fornecidas pelo software que está sendo avaliado.
3. levantar Requisitos de Domínio: os requisitos de domínio dizem respeito aos aspectos pedagógicos/cognitivos importantes para o aprendizado do conteúdo abordado pelo

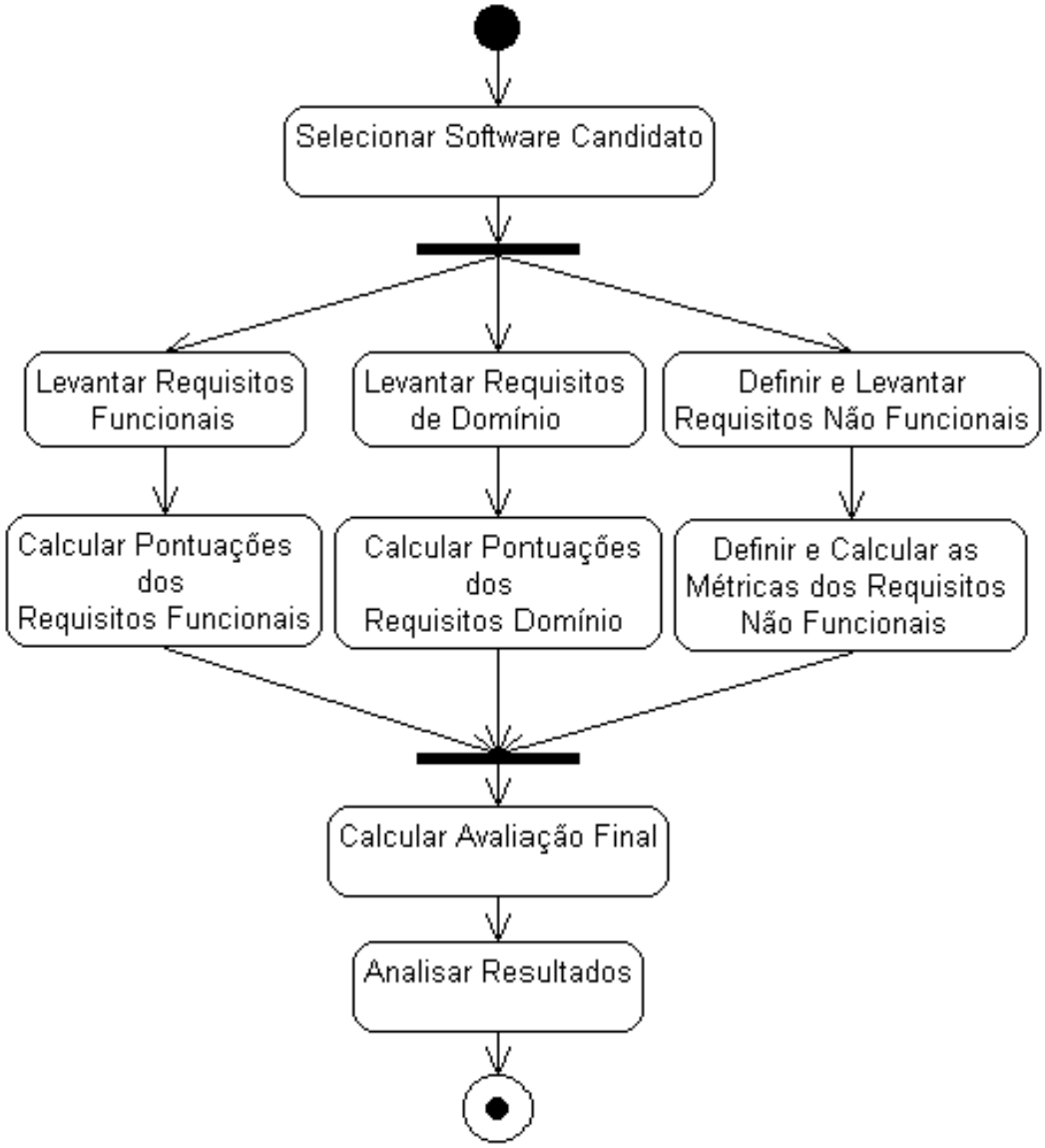


Figura 4: Atividades da Metodologia de Avaliação.

software. Para realizar essa atividade utilizamos a Teoria dos Campos Conceituais de Vergnaud, de modo a considerarmos mais precisamente os esquemas necessários para a aprendizagem dos conceitos matemáticos trabalhados no software. Essa atividade é realizada mediante o preenchimento de um formulário de avaliação de requisitos de domínio, que denominamos de Tabela de Avaliação de Requisitos de Domínio.

4. definir e levantar Requisitos Não Funcionais: nessa etapa são identificados os requisitos não funcionais que devem ser utilizados para avaliar o software. Esses requisitos devem ser escolhidos observando-se os objetivos definidos para o software no contexto dos conteúdos que estão sendo utilizadas na sala de aula.

5. calcular Pontuações dos Requisitos Funcionais: depois de devidamente especificados os casos de uso dos requisitos funcionais, realizamos o cálculo das pontuações desses requisitos. Esse cálculo é fornecido aplicando a técnica de pontos de casos de uso. Como resultado dessa atividade, teremos o valor funcional estimado do software.

6. calcular Pontuações dos Requisitos de Domínio: os requisitos pedagógicos são calculados nessa atividade através de cálculos realizados baseado nos valores da Tabela de Avaliação de Requisitos de Domínio. Como resultado dessa atividade, teremos o valor pedagógico/cognitivo estimado do software.

7. definir e Calcular as Métricas dos Requisitos Não Funcionais: baseado nos requisitos não funcionais levantados na atividade 4 devemos realizar a definição e o cálculo das métricas utilizadas para aferir os requisitos não funcionais escolhidos.

8. calcular Avaliação Final: o cálculo final da avaliação do SE é realizado utilizando os valores levantados nas atividades anteriores. Esse valor final será composto pelos fatores funcionais, não funcionais e por requisitos pedagógicos, permitindo que seja realizada uma avaliação mais precisa por parte do professor.

9. analisar Resultados: com os resultados das avaliações de SE de mesmo domínio, o avaliador poderá escolher os produtos de software que venham melhor se adequar às métricas que foram aferidas.

## 5.2 Especificação de Casos de Uso e Cálculo de Métricas Funcionais

Seguindo a classificação dada por OLIVEIRA, GOMES e BORGES NETO (2001) no Capítulo 2 e observando que os requisitos funcionais dizem respeito às funções realizadas pelo soft-

ware em resposta a um estímulo do usuário, consideramos neste trabalho que em alguns tipos de produtos de software esses requisitos são desenvolvidos em termos pedagógicos, o que o enquadraria como um requisito de domínio pedagógico. É o caso de alguns jogos e ambientes de simulação, que não obrigatoriamente possuem atividades funcionais bem dimensionadas e descontextualizadas das ações pedagógicas.

Para os demais tipos de produtos de software, as ações dos usuários são estímulos para a realização de ações. Nesse caso, fica clara a definição de requisitos funcionais e, dessa forma, esses podem ser capturados, utilizando casos de uso. Seria o caso de produtos de software do tipo exercício-prática. Nesses produtos, os requisitos funcionais podem ser elicitados de acordo com as práticas que estão sendo desenvolvidas.

Neste trabalho, para a especificação dos casos de usos dos SE algumas restrições devem ser observadas. Essas restrições tanto dizem respeito aos tipos de atores que podem fazer parte dos cenários quanto à definição da importância dos atores e dos casos de uso. Essa importância será utilizada para a definição de pontuações para o cálculo dos pontos de casos de uso para o software. Dessa forma, temos as seguintes restrições:

1. devem ser utilizados apenas dois atores (Professor, Aluno), com pesos 2 e 3 respectivamente. Essa restrição mostra que estamos valorizando apenas os papéis dos professores e alunos para o software que está sendo avaliado. Os pesos para cada ator são definidos segundo a importância da cada um no processo. Nesse caso, propomos a definição de um peso mais alto para o aluno, de modo a priorizarmos a escolha de produtos de software que ofereçam características funcionais direcionadas a esse ator. Contudo, também pontuamos positivamente a participação do professor como mediador na utilização de um SE, como apresentado em OLIVEIRA, GOMES e BORGES NETO (2001).

2. os casos de uso devem possuir seus pesos definidos de acordo com a sua importância para o tópico de estudo, variando de 0 a 5: para um software de ensino de equações para turmas iniciais, a apresentação dos passos do cálculo pode ser pontuado com 5. Já esse mesmo fator não é relevante para um software de nível superior, podendo ser pontuado com um, ou até mesmo zero.

Não utilizaremos nesse momento, fatores de ajuste para os pontos de casos de uso. Esses fatores normalmente se referem a requisitos não funcionais do software e, no nosso caso, esses requisitos serão definidos em uma etapa subseqüente. Desse modo, o cálculo final para as métricas dos requisitos funcionais dos pontos de casos de uso será composto apenas pelas *Unadjusted Use Case Points* (UUCP), seguindo a fórmula apresentada anteriormente:

$$UUCP = UAW + UUCW$$

## 5.3 Levantamento e Cálculo de Pontuações de Requisitos de Domínio

Os requisitos de domínio, no caso deste trabalho, são requisitos relativos a fatores pedagógicos/cognitivos. Abordamos esse tema utilizando a Teoria dos Campos Conceituais de Vergnaud em que, um campo conceitual pode ser definido como: um conjunto de situações que dão sentido a um conceito; um conjunto de invariantes e um conjunto de representações simbólicas para o conceito. Se utilizarmos o campo conceitual da álgebra, esses fatores podem ser representados da seguinte forma:

- situações: aqui devem ser relacionados as situações e os problemas a resolver, visto que esses fatores é que dão sentido ao conceito que está sendo tratado. Por exemplo, calcular a área de um quarto com o propósito de revesti-lo;
- invariantes: são os princípios lógicos subjacentes ao próprio conceito. No caso da álgebra, podem ser considerados como invariantes: incógnita, variável, equação, inequação, fórmula e função;
- representações: é a forma como os invariantes podem ser representados de modo a permitir a manipulação dos conceitos. Essas representações podem ser formais, tais como símbolos de igualdade (= ), ou informais, tais como figuras e objetos (representação icônica).

Para a realização do levantamento dos requisitos de domínio, utilizamos nesse trabalho, uma tabela, denominado Tabela de Avaliação de Requisitos de Domínio. Nessa tabela, observaremos os três aspectos definidos por Vergnaud para a construção de um conceito, sendo que são pontuados positivamente os produtos de software que utilizarem essas representações. Além disso, pontuamos a contextualização das representações com relação ao perfil dos alunos que utilizarão o software e pela inter-relação entre os aspectos. Assim, fatores como a utilização de representações icônicas para turmas iniciais é pontuada positivamente.

A Tabela 1 deve ser utilizada para a realização do levantamento de requisitos de domínio para o campo conceitual da álgebra.

Para realizar o cálculo de pontuações para os requisitos de domínio, consideraremos esses tipos de requisitos como fatores técnicos (*Technical Factors*) dos pontos de casos de uso. Assim,

Tabela 1: Requisitos de Domínio.

| | |
|---|---|
| Nível de Ensino Abordado | [ ]Ensino Infantil<br>[ ] Ensino Fundamental<br>[ ] Ensino Médio<br>[ ] Ensino Superior<br>[ ] Não Se Aplica |
| Situações Apresentadas | [ ] Problemas de Equações<br>[ ] Problemas de Inequações<br>[ ] Funções<br>[ ] Sistemas de Equações<br>[ ] Sistemas de Inequações<br>[ ] Outro _____<br>[ ] Não Apresenta situação |
| Invariantes Envolvidas | [ ] Variáveis<br>[ ] Incógnitas<br>[ ] Equações<br>[ ] Inequações<br>[ ] Funções<br>[ ] Fórmulas<br>[ ] Outro _____ |
| Representações Utilizadas | [ ] Formal<br>[ ] Linguagem Natural<br>[ ] Icônica<br>[ ] Outro_____ |
| Correlações | [ ]Representações são compatíveis com o nível de ensino abordado ?<br>[ ] Existem relações diretas observavéis entre as Representações e os Invariantes? |

Tabela 2: Pesos dos Fatores Técnicos.

| Fator | Peso |
|---|---|
| Situação | 5 |
| Invariantes | 2 |
| Representações | 1.5 |
| Correlações | 5 |

Tabela 3: Avaliação de Fatores Técnicos.

| Fator | Cálculo dos Valores |
|---|---|
| Situação | Adicionar 1 ponto para cada Situação selecionada na Tabela 1. Usar "0" se o item escolhido for "Não Apresenta Situação" |
| Invariantes | Adicionar 1 ponto para cada Invariante selecionada na Tabela 1 |
| Representações | Adicionar 1 ponto para cada Representação selecionada na Tabela 1 |
| Correlações | Adicionar 2 pontos para cada Correlação selecionada na Tabela 1 |

utilizaremos a Tabela 2 de pesos para os fatores. Na Tabela 3, apresentamos a forma como se avaliar cada um dos fatores no software.

Como apresentado no Capítulo 4, o cálculo final dos fatores técnicos relacionados aos requisitos de domínio levantados a partir de uma perspectiva dos campos conceituais de Vergnaud, deve ser realizado utilizando a seguinte fórmula:

$$TCF = 0.6 + (.01 * TFactor)$$

Onde TCF é o Fator de Complexidade Técnica (*Technical Complexity Factor*) e Tfactor é o Fator Técnico Total (*Total Technical factor*) que é calculado pela soma dos produtos dos pesos de cada fator técnico pelo valor levantado pela Tabela 2.

Os valores definidos para os pesos na Tabela 2, seguem a variação sugerida em KARNER (1993), que foi definida de 0 a 5. No nosso caso, como estamos realizando a avaliação de requisitos de domínio e, nesse caso, o que é mais importante são os fatores pedagógicos subseqüentes a esse conceito, avaliamos com valores maiores os SE que ofereçam situações que forneçam o aprendizado de um conceito e os que possuem correlações entre essas situações e sua representações. Por esse motivo, o fator Situação e Correlações possuem peso 5.

Tabela 4: Pesos dos Fatores Ambientais.

| Requisito Não Funcional | Peso |
|---|---|
| Tempo para completar uma tarefa | 5 |
| Razão entre sucessos e falhas | 2 |
| Frequência da utilização do help ou da documentação | 2 |
| Número de clicks do mouse | 3 |
| Distância percorrida pelo mouse | 3 |

## 5.4 Definição, Levantamento e Cálculo de Pontuações de Requisitos Não Funcionais

Na fase de definição dos requisitos não funcionais, escolhemos que requisitos não funcionais são importantes para os produtos de software que estão sendo avaliados. Por questão de simplificação, será utilizada neste trabalho apenas a usabilidade como requisito não funcional a ser avaliado. Para o requisito de usabilidade devem ser definidas e especificadas as métricas a serem utilizadas. Segundo NIELSEN (2001), particularmente para usabilidade podemos utilizar métricas como:

1. tempo para completar uma tarefa;
2. razão entre sucessos e falhas;
3. freqüência da utilização do help ou da documentação;
4. número de clicks do mouse;
5. distância percorrida pelo mouse.

Para definir a importância dessas métricas em relação à avaliação que estamos realizando, devemos fornecer um peso para cada métrica. Na metodologia que estamos propondo, cada métrica de requisitos não funcionais está sendo tratada como um fator ambiental (*Environment Factor*) definido na técnica de pontos de casos de uso. Na Tabela 4, apresentamos as pontuações para esses requisitos:

Para realizar o levantamento dos requisitos não funcionais nos produtos de software que estão sendo avaliados temos que utilizar cada software e anotar as informações necessárias para cada requisito. Para automatizar essa tarefa apresentamos em SOUZA, CASTRO FILHO e PEQUENO (2004), uma ferramenta para o suporte a extração de requisitos não funcionais de SE.

Como apresentado no capítulo 4, o cálculo do valor relativo aos requisitos não funcionais é realizado pela fórmula seguinte, que é utilizada para o cálculo de fatores ambientais dos pontos de casos de uso:

$$EF = 1.4 + (-0.03 * EFactor)$$

Onde EF significa fator ambiental (*Environment Factor*) e EFactor significa Fator Ambiental Total (*Total Environment Factor*).

Os valores para cada requisito a ser elicitado nos produto de software, devem variar entre 0 e 5. Esses valores devem ser ponderados de acordo com os resultados mais elevados, e ajustados para que fiquem dentro da faixa de variação. Por exemplo, se for avaliado um requisito não funcional entre os produtos que estão sendo comparados tendo 20 como seu maior valor, esse deve ser considerado como 5 e, utilizando uma regra de três simples, todos os demais valores devem ser ajustados.

## 5.5 Cálculo da Avaliação Final e Análise de Resultados

Depois de realizar todos os levantamentos de requisitos funcionais, de domínio e não funcionais, a atividade seguindo do processo de avaliação é calcular o valor final do software. Esse valor final, também utilizando uma abordagem baseada em pontos de casos de uso, é fornecido pela fórmula:

$$AUCP = UUCP * TCF * EF$$

Onde AUCP significa Pontos de Casos de Uso Ajustados (*Adjusted Use Case Points*). Utilizando esses valores calculados para cada software candidato, podemos ter uma avaliação precisa de diversos aspectos do software, desde fatores puramente operacionais até fatores pedagógicos/cognitivos.

Explicado o funcinamento da Técnica de Pontos de Casos de Uso e da Teoria dos Campos Conceituais em nossa metodologia, resolvemos apresentar no capítulo a seguir um estudo de caso no intuito de validá-la.

# 6 Estudo de Caso

Para validar a abordagem apresentada neste trabalho, apresentamos um exemplo simplificado da avaliação comparativa de dois SE utilizados para auxiliar no ensino de álgebra.

## 6.1 Seleção dos Produtos de Software Candidatos

Os produtos de software escolhidos para apresentar a nossa abordagem de avaliação comparativa são dois produtos de software que apresentam características funcionais e não funcionais bastante semelhantes, ou seja, possuem a mesma área de aplicação, o que a priori seria bastante difícil para o professor decidir qual software usar em suas aulas.

Ambos os software possuem como área de aplicação a álgebra, mais especificamente o ensino de equação, fazendo uso de uma balança de dois pratos como forma de contextualizar a introdução do conceito para os alunos de 5a a 8a série.

O primeiro software utilizado foi o Jogo da Balança de LUCA (1995). Esse software permite, através de uma balança de dois pratos, trabalhar com o conceito de equações e comparação entre quantidades (Figura 5). Desenvolvido em sete níveis de dificuldade, o software permite a utilização de representações gráficas de maçãs, carneiros, entre outras, para mostrar a igualdade entre quantidades nos dois pratos da balança. O aluno, através de operações de adição, subtração, multiplicação e divisão, pode manipular as quantidades nos pratos até conseguir o equilíbrio. Nos níveis mais avançados, podem ser utilizados pesos para manipular quantidades maiores.

O segundo software utilizado foi o Balança Interativa de CASTRO FILHO (2004). Esse software também trabalha com uma balança de pratos e, como no Jogo da Balança, trabalha com os conceitos de equações. Além disso, o software também trabalha com conceitos de inequações, permitindo a comparação entre quantidades diferentes e mostrando a noção de maior que e menor que. Desenvolvido em dez níveis diferentes, esse software permite a utilização de pesos com valores desconhecidos (representados por letras) e de pesos com valores conhecidos,

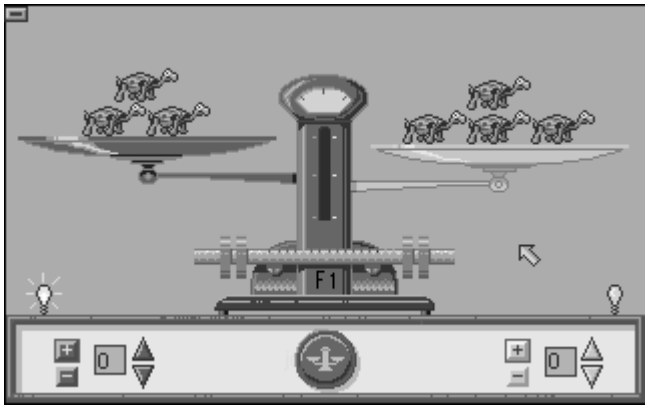


Figura 5: Jogo da Balança.

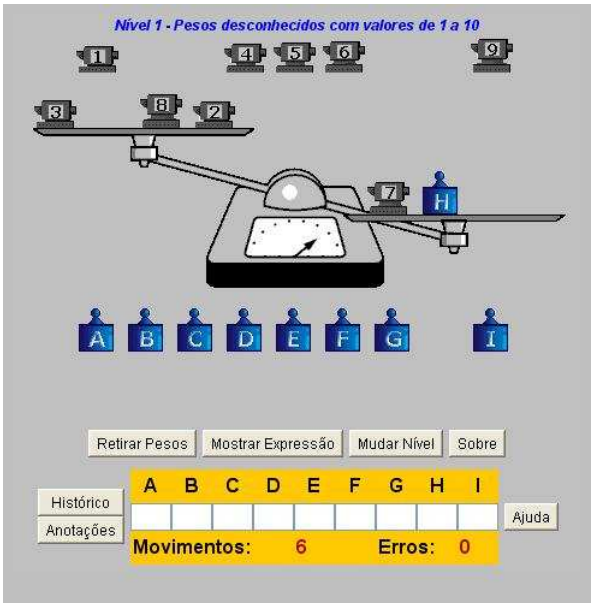


Figura 6: Balança Interativa.

sendo que o aluno deve, de um lado da balança, colocar os pesos desconhecidos e do outro os pesos conhecidos para compará-los, e descobrir o valor dos pesos desconhecidos (Figura 6). Nesse software, a expressão matemática referente à disposição de pesos poderá ser apresentada em qualquer tempo. Em níveis mais elevados, essas expressões podem ser manipuladas diretamente.

## 6.2 Especificação de Casos de Uso e Cálculo de Métricas Funcionais

Os dois produtos de software que estão sendo avaliados foram desenvolvidos para serem utilizados pelos alunos, com a mediação do professor, para o auxílio do desenvolvimento de

conceitos algébricos. Contudo, para o cálculo do valor dos requisitos funcionais para esses produtos, apenas o ator Aluno será considerado, visto que o ator professor não irá realizar tarefas diretas no software.

Inicialmente, para o cálculo do valor dos requisitos funcionais do Jogo da Balança, podemos identificar os seguintes casos de uso com seus respectivos pesos, apresentado na Tabela 5.

Tabela 5: Valores para o Software Jogo da Balança.

| Casos de Uso | Peso |
|---|---|
| Escolher Pesos | 3 |
| Exibir Expressão Algébrica | 3 |
| Mudar Nível | 2 |

Já a avaliação de requisitos funcionais para o Balança Interativa foi considerada utilizando os seguintes casos de uso com seus respectivos pesos, apresentado na Tabela 6.

Tabela 6: Valores para o Software Balança Interativa.

| Casos de Uso | Peso |
|---|---|
| Escolher Pesos | 5 |
| Exibir Expressão Algébrica | 5 |
| Mudar Nível | 2 |
| Limpar Pratos | 0 |

Na utilização do Jogo da Balança, os pesos são gerados dos dois lados dos pratos. Em seguida, é permitida a realização de operações de adição ou subtração de unidades em cada lado do prato. Então podemos verificar se os pratos estão equilibrados através de um botão que mostra a expressão matemática correspondente.

No Balança Interativa, os pesos podem ser escolhidos livremente dos dois lados dos pratos, sendo que é possível utilizar pesos com valores desconhecidos (representados por letras) ou pesos com valores conhecidos. Em seguida, o aluno pode manipular esses pesos e trabalhar com equilíbrio entre os pratos, substituindo os pesos livremente. Um outro fator importante é a utilização de um botão que pode ser acionado em qualquer tempo para apresentar a expressão algébrica formal relativa à configuração da balança.

Por conta de uma maior flexibilidade na escolha dos pesos, a Balança Interativa tem esse requisito com um valor maior que o correspondente no Jogo da Balança. Já pelo fato de ser permitida a verificação das expressões algébricas formais a qualquer tempo, o Balança Interativa também tem esse requisito mais bem avaliado que no Jogo da Balança. Desse modo, são calculados os pontos de casos de uso para cada software da seguinte forma:

Jogo da Balança:

UAW=3 * 1=3

UUCW=2*3+1*2=8

UUCP=UAW+UUCW=11

Balança Interativa:

UAW=3*1=3

UUCW=2*5+1*2+1*0=12

UUCP=UAW+UUCW=15

Até aqui, podemos concluir que, funcionalmente, o Balança Interativa fornece mais recursos para o aluno. Isso implica dizer que o Balança Interativa realiza melhor as tarefas que elicitamos nos casos de uso de acordo com os valores de importância que demos para cada caso de uso.

## 6.3 Levantamento e Cálculo de Pontuações de Requisitos de Domínio

Para um software de apoio ao aprendizado de álgebra, além dos fatores funcionais, também os fatores pedagógicos subjacentes aos conceitos devem ser tratados de alguma forma. Nesse sentido, abordamos a avaliação desse tipo de requisito, aqui tratado com requisito de domínio, utilizando a Teoria dos Campos Conceituais de Vergnaud.

Aplicando a Tabela de Avaliação de Requisitos de Domínio (Tabela 1), obtivemos os seguintes resultados, apresentados nas tabelas 7 e 8.

Tabela 7: Resultados para o Software Jogo da Balança.

| Fator | Peso | Valor Levantado | Resultado |
|---|---|---|---|
| Situação | 5 | 1 | 5 |
| Invariantes | 2 | 1 | 2 |
| Representações | 1.5 | 2 | 3 |
| Correlações | 5 | 2 | 10 |
| | | | 20 |

Tabela 8: Resultados para o Software Balança Interativa.

| Fator | Peso | Valor Levantado | Resultado |
|---|---|---|---|
| Situação | 5 | 1 | 5 |
| Invariantes | 2 | 3 | 6 |
| Representações | 1.5 | 3 | 4.5 |
| Correlações | 5 | 4 | 20 |
| | | | 35.5 |

Na avaliação sobre Situação, ambos os produtos trabalham com a abstração de uma balança para trabalhar com os conceitos algébricos. Dessa forma, ambos foram avaliados da mesma maneira.

Com relação aos Invariantes, os dois produtos trabalham com a noção de equações através da abstração do equilíbrio entre os pratos. O Balança Interativa utiliza também as noções de inequações, através da manipulação dos pesos, enquanto a balança está em desequilíbrio, e de incógnitas, com a utilização de pesos de valores desconhecidos.

É utilizada uma representação baseada em figuras (balança, pesos, maçãs, tartarugas, dentre outros) para manipular as informações de Invariantes utilizadas nos dois produtos, bem como a representação formal para as equações. Adicionalmente, o Balança Interativa, faz uso de representações formais para incógnitas e inequações.

As correlações observadas nos dois produtos indicam que suas representações são compatíveis com os níveis educacionais onde eles são aplicados. Já o Balança Interativa apresenta uma melhor relação às representações e os invariantes, permitindo a visualização direta dos conceitos de inequações e incógnitas.

O cálculo final dos resultados da avaliação de requisitos de domínio é o seguinte:

Jogo da Balança:

TCF=0.6+(.01*20)=0.6+0.2=0.8

Balança Interativa:

TCF=0.6+(.01*35.5)=0.6+0.355=0.95

Podemos concluir com a avaliação dos requisitos de domínio que o Balança Interativa apresenta uma melhor avaliação com relação a fatores pedagógicos.

## 6.4 Definição, Levantamento e Cálculo de Pontuações de Requisitos Não Funcionais

Para a avaliação dos requisitos não funcionais, realizamos a tarefa de verificar uma equação simples (ex.: 2+X=5 ). No Jogo da Balança, essa tarefa é realizada utilizando as operações de adição ou subtração para tentar igualar a quantidade de elementos nos pratos da balança. Assim, é escolhida uma operação e então é definida a quantidade de elementos que deverão ser adicionados ou subtraídos (dependendo da operação escolhida), clicando nas setas próximas à caixa de texto fornecida para este fim até, atingir o valor desejado.

No Balança Interativa, existem várias incógnitas cujos valores devem ser descobertos. Para essa avaliação, utilizaremos apenas uma incógnita, de modo a ser possível realizar uma comparação mais aproximada com o Jogo da Balança. Para trabalhar uma equação simples, devemos inicialmente escolher um peso sem valor definido e colocá-lo em um dos pratos. Isso é feito, clicando sobre o peso e arrastando para o prato desejado. Em seguida, devemos selecionar os pesos com valores definidos e arrastar (ou retirá-los) para pratos até alcançar o equilíbrio.

Na definição das métricas não funcionais para a avaliação, utilizamos três métricas simples. A primeira, o número de clicks no mouse para a realização de uma única operação, cujo peso que definimos é 2, visto que esse é um critério relativamente relevante para a usabilidade de um software, principalmente para o público ao qual ele se destina.

A segunda, foi o tempo necessário para realizar uma operação simples, que recebeu peso 2,

por ser esse um fator que envolve diretamente a motivação do aluno, visto que o tempo longo na realização de uma operação pode fazer o aluno perder a atenção no conteúdo. A terceira, foi a distância percorrida pelo mouse na realização da tarefa. Essa métrica está ligada à dificuldade da operação de arrastar-e-soltar para pessoas que não utilizam computador freqüentemente ou com coordenação motora ainda em formação. Assim, esse critério foi pontuado com peso 3.

Vale ressaltar que existem diversas outras métricas que poderiam ser utilizadas, contudo resolvemos colocar neste trabalho, apenas essas três métricas, que já são suficientes para exemplificar a aplicação da metodologia proposta.

Os resultados obtidos na avaliação estão apresentados nas tabelas 9 e 10.

Tabela 9: Resultados para o Software Jogo da Balança.

| Requisito Não Funcional | Peso | Valor Levantado | Resultado |
|---|---|---|---|
| Número de clicks no mouse | 2 | 5 | 10 |
| Tempo de uma operação | 2 | 3.75 | 7.5 |
| Distância percorrida pelo mouse | 3 | 2.67 | 8.01 |
| | | | 25.51 |

Tabela 10: Resultados para o Software Balança Interativa.

| Requisito Não Funcional | Peso | Valor Levantado | Resultado |
|---|---|---|---|
| Número de clicks no mouse | 2 | 0.71 | 1.42 |
| Tempo de uma operação | 2 | 5 | 10 |
| Distância percorrida pelo mouse | 3 | 5 | 15 |
| | | | 26.42 |

Na avaliação do número de clicks do mouse para a realização de uma operação simples no Jogo da Balança, é necessário 1 click para se escolher a operação, mais uma quantidade de clicks definida pela número a ser adicionado ou subtraído de um dos pratos e um outro click para se verificar o resultado. Assim, temos um total de 2 + VI (Valor da Incógnita) que estamos procurando. Para realizar essa avaliação, vamos supor que a incógnita tem valor 5, assim teremos um total de 7 clicks.

Para a avaliar o número de clicks realizados no Balança Interativa, não consideramos os

clicks necessários para selecionar os pesos. Assim, consideramos apenas um click para apresentar o resultado da expressão algébrica formal, quando atingimos o resultado desejado.

Realizando o ajuste dos valores para a faixa de aceitação (entre 0 e 5) , temos que o valor 7 passa a ser 5 e o valor 1 passa a ser 0.71.

Com relação ao tempo necessário para realizar uma operação simples no Joga da Balança, podemos verificar facilmente que isso dependerá do conhecimento do aluno na realização de operações aritméticas simples. Assim, quanto melhor o domínio do aluno dessas operações, menor será o tempo necessário para equilibrar os pratos. Isso decorre do fato de que basta apenas adicionar ou subtrair valores aos pratos para realizar a operação. Para essa avaliação, utilizamos um valor empírico de 15s.

Já para realizar a avaliação do tempo necessário para a realização de uma operação simples no Balança Interativa não podemos considerar somente o conhecimento em aritmética. Nesse software, utilizamos incógnitas (representados por letras) que devemos tentar equilibrar, usando os pesos com valores conhecidos. Assim, a sorte pode ser também considerada um fator importante, visto que o equilíbrio é conseguido observando-se os valores dos pesos que escolhemos aleatoriamente. Também utilizamos um valor empírico médio de 20s para avaliar esse requisito.

Realizando o ajuste nos valores, considerando a escala de 0 a 5, temos que o valor 20 será avaliado como 5 e o valor 15 será avaliado como 3.75. O terceiro e último critério utilizado, que é a distância percorrida pelo mouse para realizar uma operação, foi aferida no Jogo da Balança, considerando as distâncias percorridas entre os botões de escolha das operações e de verificação de valor, e entre a caixa de texto de definição de quantidades. O valor médio, calculado da mesma forma que no requisito anterior, foi aferido nesse software como 24cm. No Balança Interativa, pelo fato de todos os pesos serem manipulados através de operações de arrastar-e-soltar, esse requisito teve seu valor médio aferido como 45cm. Esses valores foram encontrados utilizando-se um monitor de 15" com resolução de 1024x768 pixels.

Dessa forma, realizando o ajuste nos valores considerando a escala de 0 a 5, temos que o valor 45 será avaliado como 5 e o valor 24 será avaliado como 2.67.

O resultado final dos cálculos para os requisitos não funcionais é o seguinte:

Jogo da Balança:

EF=1.4+(-0.03*25.51)=1.4-0.7653=0.6347

Balança Interativa:

EF=1.4+(-0.03*26.42)=1.4 0.7926=0.6074

Os resultados obtidos na avaliação de requisitos não funcionais nos levam a concluir que, com relação a esse tipo de requisito, o Jogo da Balança leva vantagem sobre o Balança Interativa. Observamos ainda que nos cálculos dos fatores ambientais (EF), quanto maior os valores não funcionais aferidos, menor será o resultado da avaliação. Isto ocorre devido aos valores que são multiplicados por um fator negativo (-0.03) no cálculo final das pontuações.

É válido ressaltar que os valores não funcionais aferidos estão tecnicamente ligados ao perfil do avaliador. Contudo, como as avaliações que são apresentadas nesse exemplo foram realizadas por um mesmo avaliador, e como temos a intenção de comparar os produtos de software, as aproximações que fizemos não afetam diretamente os resultados.

## 6.5 Cálculo da Avaliação Final e Análise de Resultados

Com base nos resultados dos requisitos funcionais, de domínio e não funcionais, podemos efetuar os cálculos necessários à obtenção dos resultados finais da avaliação para os dois produtos de software.

Esses cálculos são realizados da seguinte maneira:

Jogo da Balança:

AUCP=11*0.8*0.6347=5.58536

Balança Interativa:

AUCP=15*0.95*0.6074=8.65545

Esses resultados nos mostram que, de acordo com a nossa metodologia, o produto de software Balança Interativa é melhor do que o produto de software Jogo da Balança.

Somente essa informação já seria útil para a realização da seleção desse software para ser utilizado em sala de aula. Contudo, a nossa metodologia permite uma visualização mais

ampla dos aspectos relevantes à tarefa de seleção de um SE. De fato, se observarmos os valores finais calculados, podemos perceber que ele é uma composição de resultados. Cada um desses resultados já pode, por si só, fornecer informações importantes para a seleção do SE ou seja, o software poderá ser selecionado levando-se em consideração apenas aspectos funcionais, não-funcionais ou de domínio de forma isolada. Assim, podemos priorizar os requisitos de domínio (pedagógico-cognitivos) na seleção, sendo que, por exemplo, podemos selecionar um SE que seja rejeitado pelo cálculo final das pontuações e que tenha um valor de requisitos de domínio superior.

Com base na metodologia apresentada e com o intuito de torná-la viável a qualquer usuário, resolvemos propor uma ferramenta que automatizasse todo o processo que essa metodologia incorpora. Essa ferramenta tem o intuito de facilitar o uso dessa metodologia por qualquer usuário, ou seja, os conhecimentos de engenharia de software e de Teoria dos Campos Conceituais, não serão obstáculos para os leigos no assunto, caso queiram fazer uso dessa ferramenta.

Dessa forma, resolvemos apresentar no capítulo a seguir a ferramenta que irá automatizar esse processo de avaliação bem como suas interfaces.

# 7 Ferramenta de Avaliação Automática de SE (FASE)

## 7.1 Introdução

Neste capítulo, apresentaremos o protótipo de uma ferramenta que implementa a metodologia de avaliação comparativa de software educativo, proposta no Capítulo 5. Esta ferramenta, denominada FASE (Ferramenta de Avaliação Automática de Software Educativo), permite que o professor realize avaliações de diversos produtos de SE sem a necessidade de entender explicitamente os conceitos de engenharia de requisitos nem de aspectos cognitivos, que são embutidos na metodologia de avaliação implementada pela ferramenta, visto que os termos técnicos utilizados pela ferramenta são apresentados de forma implícita em FASE, possibilitando a utilização de uma linguagem menos técnica pelo professor/avaliador na realização do processo de avaliação de seus produtos de software.

Nesse capítulo a FASE é apresentada considerando a ordem com que as suas interfaces são disponibilizadas para o avaliador. Assim, nas seções seguintes, descrevemos o funcionamento dessas interfaces.

## 7.2 FASE (Ferramenta de Avaliação Automática de SE)

O objetivo principal da FASE é permitir que um professor seja capaz de realizar uma escolha mais minuciosa de quais softwares educativos utilizar em sala de aula para uma determinada área de aplicação da matemática (Campo Conceitual).

Nessa ferramenta, não são levados em consideração para aferição de um software apenas aspectos relacionados à qualidade da interface gráfica, que em muitos casos acabam escondendo outros aspectos importantes com relação a usabilidade do software, mas também permitirá se aferir outras qualidades de um produto através dos seus requisitos funcionais, considerando as funcionalidades oferecidas e de seus requisitos de domínio, que dizem respeito a fatores

cognitivos/pedagógicos.

Essa ferramenta irá permitir ao avaliador/professor selecionar esses requisitos de maneira transparente, através de algumas perguntas pertinentes aos SE que estão sendo avaliados. Essas perguntas serão feitas de maneira isolada para cada requisito a ser avaliado. Podendo o avaliador/professor, dependendo de seu entendimento sobre o processo de avaliação de SE, incrementar as avaliações adicionando novos requisitos que serão considerados e incorporados à FASE durante esse processo.

É valido ressaltar que essa ferramenta deverá ser utilizada paralelamente ao software que está em avaliação. Em outras palavras, FASE permitirá ao avaliador registrar as informações sobre os softwares a medida em que esses estão sendo utilizados, sendo que no final será apresentado o resultado da avaliação realizado pela ferramenta.

### 7.2.1 Tela Principal

Nessa versão de FASE as avaliações dos softwares são realizadas de forma independente, ou seja, cada software é avaliado separadamente e seus resultados são apresentados de maneira isolada, ficando a cargo do avaliador realizar a escolha dos produtos considerando esses resultados.

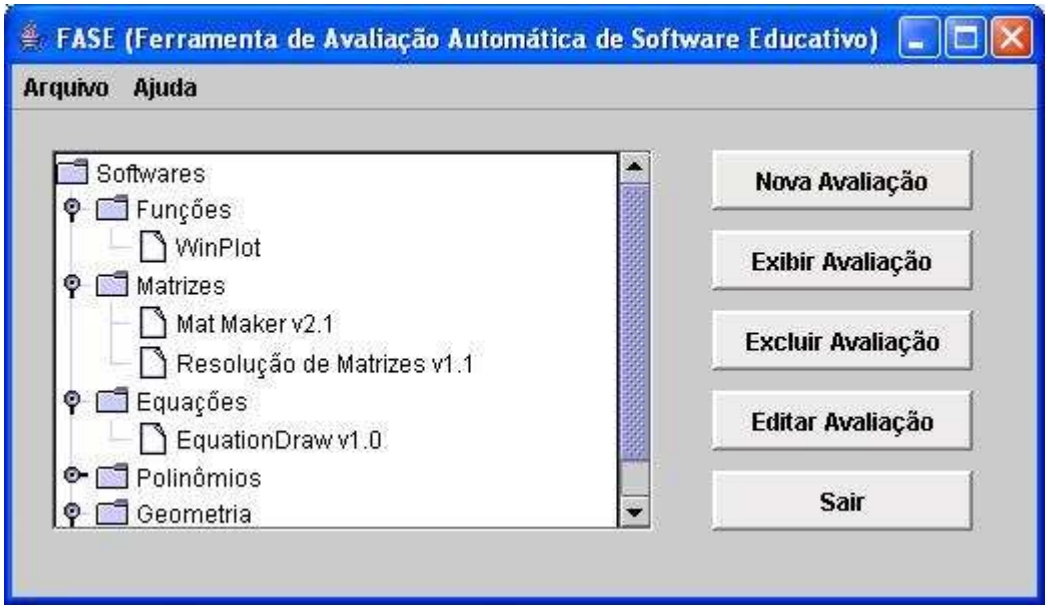


Figura 7: Tela Principal da FASE.

A Figura 7 apresenta a Tela Principal da ferramenta (FASE). Nessa tela, serão apresentados todos os softwares que já foram avaliados anteriormente, podendo o usuário selecionar um

dos softwares na lista e utilizar os botões para exibir sua avaliação, exclui-lo da lista ou ainda modificar a avaliação realizada. Além disso, o usuário poderá realizar uma nova avaliação, em que deverão ser fornecidos os dados gerais relativos ao software a ser avaliado.

### 7.2.2 Dados Gerais de Avaliação

Na Figura 8, apresentamos a tela de cadastro de dados gerais sobre o software que está em avaliação. Assim, devem ser fornecidas informações como: nome do software, versão, data da avaliação, nome do avaliador e área de aplicação do software.

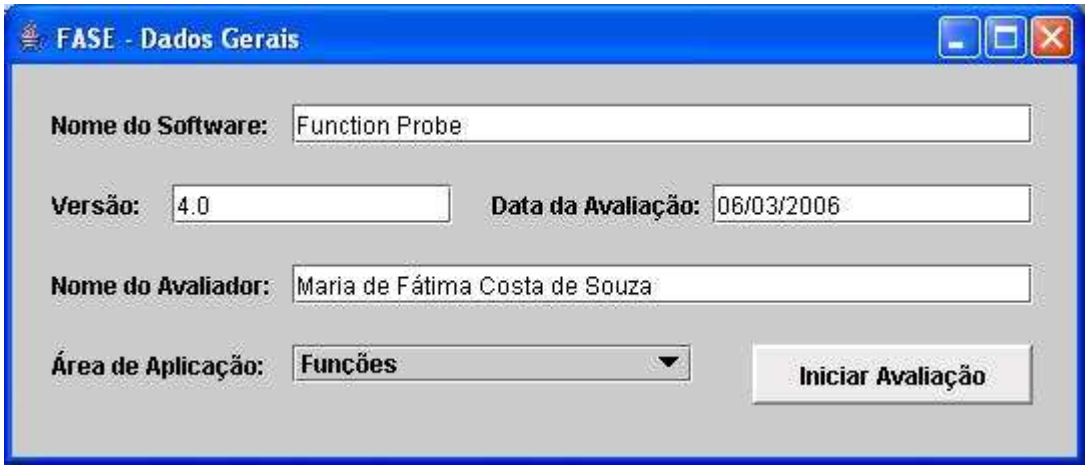


Figura 8: Dados Gerais de Avaliação.

Após o fornecimento dos dados do software que está em avaliação, o avaliador deverá dar início ao processo de avaliação pressionando o botão “Iniciar Avaliação”, que dará início a avaliação dos aspectos funcionais do software.

### 7.2.3 Avaliação Funcional

Por se tratar de um termo específico da engenharia de software, requisito funcional não poderá ser tratado, ou melhor, mencionado dessa forma em nossa ferramenta. Como o próprio nome diz, requisitos funcionais são declarações de funcionalidades que o sistema deve fornecer, como o sistema deve agir a entradas específicas e como deve se comportar em determinadas situações, ou em alguns casos, podem também declarar explicitamente o que o sistema não deve fazer. Diante dessa definição, resolvemos traduzir todas essas informações técnicas de maneira que essa ferramenta seja viável a qualquer avaliador, isto é, o conceito de requisito funcional não será obstáculo para que a avaliação seja feita.

Para tornar a atividade de definição de requisitos funcionais, que são na verdade casos de uso do sistema em questão, fornecemos para cada área de atuação dos SE (funções, matrizes, equações etc.) um conjunto mínimo de funcionalidades que podem ser utilizadas no processo de avaliação. Além disso, permitimos que o avaliador possa inserir novas funcionalidades que, na sua opinião, sejam importantes a serem levantadas. Na Figura 9, apresentamos a tela de avaliação de funcionalidades da FASE. Observe que na Figura 9 já apresentamos algumas funcionalidades, sendo que a primeira delas já é padrão para a área de funções e as demais foram adicionadas pelo avaliador, através do botão "Adicionar", para realizar a avaliação de um software que possui uma balança para trabalhar com as noções de equações, ou seja, essas funcionalidades são específicas para esse software.


Figura 9: Características Funcionais.

Tivemos cuidado em tomar essa medida de fornecer um conjunto mínimo de funcionalidades por área de aplicação, visto que o nosso intuito é permitir que todo e qualquer avaliador seja capaz de manusear a nossa ferramenta e não somente aqueles que tivessem o conhecimento em engenharia de requisitos.

Adicionalmente, para cada funcionalidade listada na tabela da Figura 9 devemos pontuar cada uma com valores entre 0 a 5, de acordo com o grau de importância que o avaliador atribuirá a funcionalidade (requisito). Sempre levando em consideração que 0 representa o requisito de menor importância e 5 o requisito de maior importância. Esses valores são utilizados na nossa metodologia de avaliação para classificar e pontuar os casos de usos, de forma a se aplicar à técnica de pontos de caso de uso para gerar o valor da avaliação desse software.

Para melhor exemplificar a utilização dessa interface, suponha que queiramos avaliar um software na área de Funções. Assim, as possíveis funcionalidade referente à essa área de atuação

serão:

- Possui Representação Formal
- Possui Representação em forma de Tabela
- Apresenta Representação Gráfica

Para cada funcionalidade relacionada acima será solicitada uma pontuação de 0 a 5 de acordo com o grau de importância do requisito para o avaliador.

Como apresentado em SOUZA, PEQUENO e CASTRO FILHO (2005a) o cálculo referente aos requisitos funcionais é tratado dentro da nossa metodologia de avaliação, como sendo equivalente aos Pontos de Casos de Uso Não Ajustados, utilizados na técnica de pontos de casos de uso, ou seja, de posse de todas as informações referentes a esses requisitos, a FASE utilizará os valores associados a esses dados para calcular de maneira pontual o valor desse requisito.

### 7.2.4 Avaliação Pedagógica

Como explicado anteriormente, requisito de domínio é um termo próprio da engenharia de requisitos. Eles nada mais são que requisitos que definem funções específicas de determinados domínios de aplicação. Esses requisitos tanto podem ser funcionais como não funcionais. Em nossa metodologia de avaliação tratamos os requisitos de domínio como requisitos relativos a fatores pedagógicos/cognitivos, no qual fizemos uso da Teoria dos Campos Conceituais para abordá-los.

Na ferramenta de avaliação, esses requisitos são apresentados ao avaliador como um conjunto de opções onde o avaliador irá selecionar as suas situações, as suas invariantes e as suas representações (Figura 10), que são os conceitos trabalhados na Teoria dos Campos Conceituais. Essas opções foram apresentadas no Capítulo 5, através da Tabela de Requisitos de Domínio (SOUZA; PEQUENO; CASTRO FILHO, 2005a, 2005b). Nesta tabela o avaliador será capaz de fazer suas seleções, sem notar que suas escolhas, nada mais são que a seleção de requisitos de domínio. É válido ressaltar que utilizamos também a noção de correlação (SOUZA; PEQUENO; CASTRO FILHO, 2005a), que permite avaliar as relações entre os fatores relacionados na Teoria dos Campos Conceituais.

A avaliação desses requisitos dentro da FASE se dará da seguinte maneira:

- O avaliador deverá informar o nível de ensino abordado, ou seja: infantil, fundamental, médio, superior ou não se aplica.

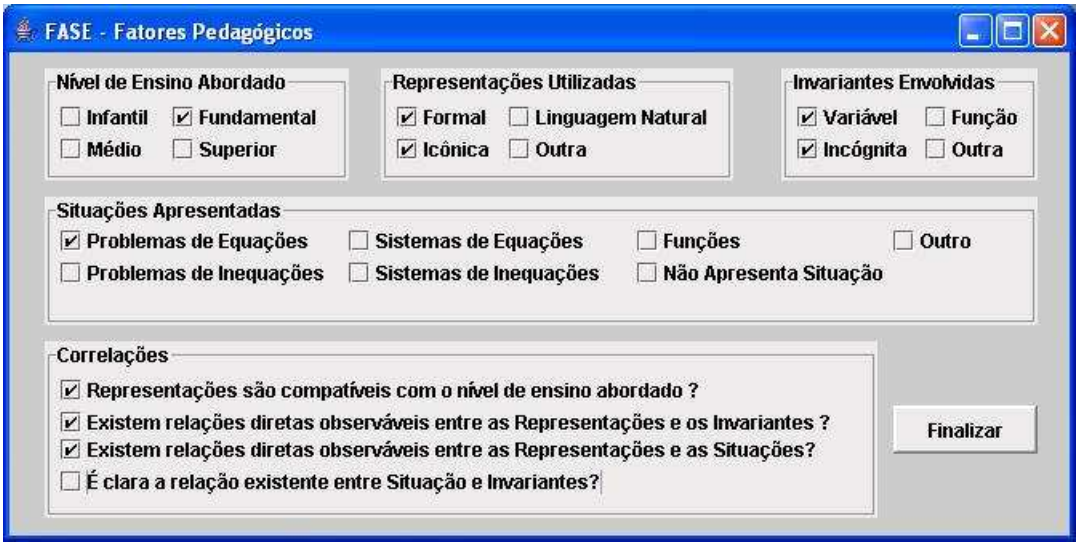


Figura 10: Fatores Pedagógicos.

- Depois, deverá selecionar a situação apresentada, ou seja: problemas de equações, problemas de inequações, funções, sistema de equações, sistema de inequações ou outro.
- Em seguida, o avaliador deverá informar o(s) Tipo(s) de invariantes envolvidas, ou seja: variáveis, incógnitas, equações, inequações, funções ou outro.
- Posteriormente, deverá selecionar qual o tipo de representação foi utilizado para as invariantes envolvidas, ou seja: formal, linguagem natural ou icônica.
- Por fim, feitas às seleções acima, o avaliador deverá observar as correlações (SOUZA; PEQUENO; CASTRO FILHO, 2005a, 2005b) existentes entre Situações, Invariantes e Representações.

Para exemplificar, podemos usar as seguintes perguntas para avaliar as correlações:

- As Representações são compatíveis com o nível de ensino abordado?
- Existem relações diretas observáveis entre as Representações e as Situações?
- Existem relações diretas observáveis entre as Representações e os Invariantes?
- É clara a relação existente entre Situação e Invariantes?

Como também apresentado em SOUZA, CASTRO FILHO e PEQUENO (2004) o cálculo referente aos requisitos de domínio é tratado dentro da nossa metodologia de avaliação, como sendo equivalente aos Fatores Técnicos, utilizados na técnica de pontos de casos de uso. Da mesma

forma apresentada para os demais tipos de requisitos, a FASE utilizará os valores associados a esses dados para também calcular de maneira pontual esse requisito.

### 7.2.5 Avaliação de Qualidade

Os fatores de qualidade dizem respeito aos requisitos não funcionais dos SE que estão sendo avaliados. Como esses requisitos devem ser avaliados através de métricas em que valores devem ser associados a cada requisito baseado na utilização do software, resolvermos, inicialmente, permitir a definição por parte do avaliador de quais serão os fatores que devem ser avaliados no software. Dessa forma, na Figura 11, apresentamos a interface de definição de Parâmetros de Qualidade.

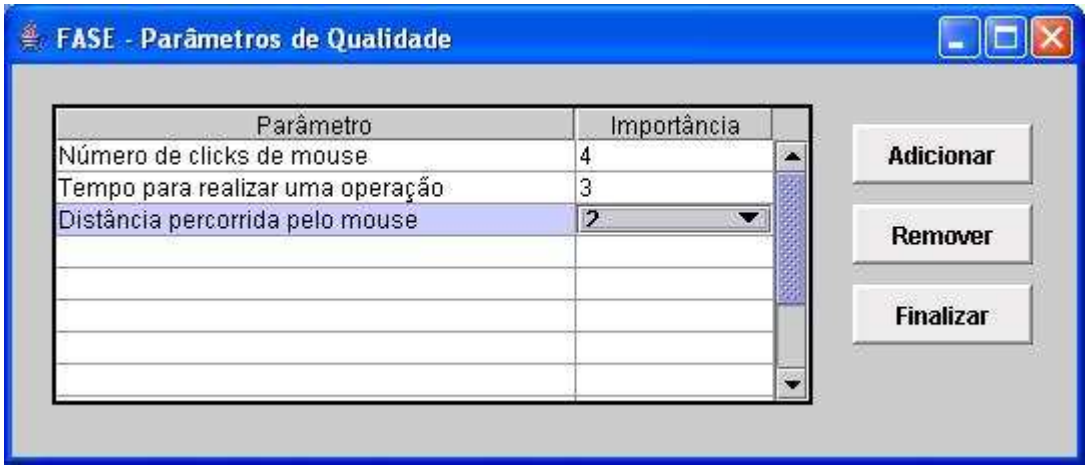


Figura 11: Parâmetros de Qualidade.

Da mesma forma que fizemos para a avaliação funcional, oferecemos para o avaliador um conjunto mínimo de parâmetros de qualidade que são gerais para qualquer tipo de software. Para exemplificar utilizamos o requisito de usabilidade. Assim, esses parâmetros, que na verdade são métricas de usabilidade, serão avaliados automaticamente quando essa interface for finalizada, através do pressionamento do botão “Finalizar”.

Para realizar a aferição de cada um dos parâmetros listados, a FASE gerará automaticamente uma tela para cada um dos parâmetros, de forma que o avaliador possa capturar os valores de cada métrica. Para exemplificar essa funcionalidade de FASE observe que o primeiro parâmetro listado é Número de clicks de mouse. Para essa funcionalidade a FASE gera uma tela, apresentada na Figura 12, onde podemos observar a presença de um botão “Iniciar” e “Finalizar”. Esses botões devem ser pressionados no início e no final, respectivamente, da realização de uma operação simples no software que está sendo avaliado. Ao pressionar o botão “Iniciar” a

FASE captura qualquer click realizado no mouse até que seja pressionado o botão “Finalizar”. Assim, essa métrica é aferida e armazenada. Essa mesma técnica é utilizada para as demais métricas definidas (parâmetros) pelo avaliador. Ou seja, ao pressionar o botão “Próximo Item” será iniciada a avaliação da próxima métrica. Isso se repete até que sejam avaliados todos os parâmetros definidos previamente.

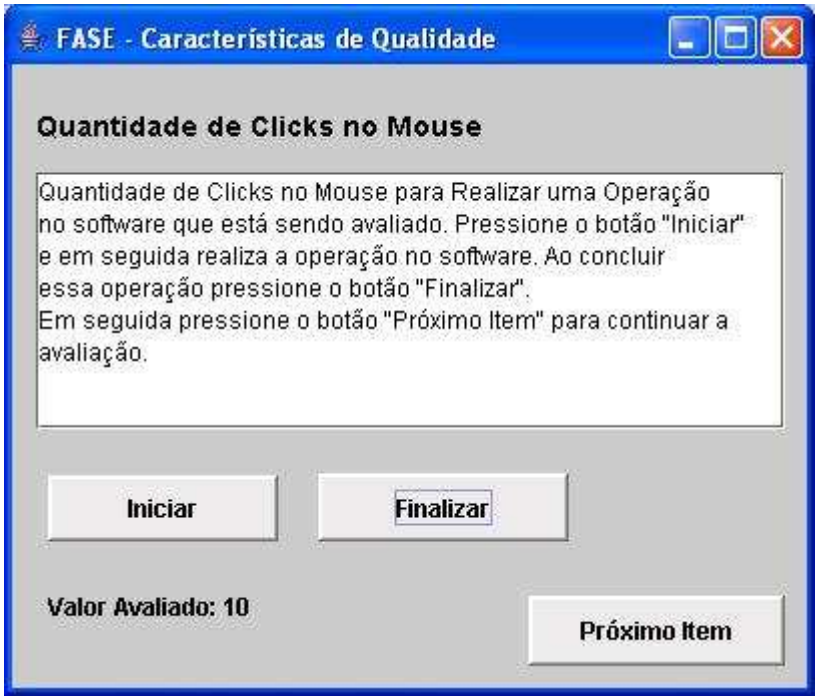


Figura 12: Aferição de Requisito Não Funcional.

Como apresentado em SOUZA et al. (2004) o cálculo referente aos requisitos não funcionais é tratado dentro da nossa metodologia de avaliação como sendo equivalente aos Fatores Ambientais, utilizados na técnica de Pontos de Casos de Uso, assim, a FASE utilizará os valores associados a esses requisitos para calcular de maneira pontual o valor dos mesmos.

### 7.2.6 Resultado Final

Já realizada a avaliação de todos os requisitos necessários, o passo seguinte e verificar o resultado final da avaliação. A Figura 13 apresenta a tela de resultado final da avaliação. Esse resultado é o produto das avaliações dos requisitos de acordo com a nossa metodologia. Uma explicação mais detalhada sobre esse cálculo e sobre a aplicação dessa técnica para a realização de avaliação de SE, pode ser encontrada no Capítulo 5 e em SOUZA et al. (2004) e SOUZA, PEQUENO e CASTRO FILHO (2005a).


Figura 13: Resultado Final de Avaliação.

## 7.3 Implementação

Atualmente já foram implementadas todas as interfaces de levantamento de informações de avaliação da FASE. A implementação da FASE está sendo realizada em Java 1.5.0 (MICROSYSTEMS, 2005) através da IDE NetBeans 5.0 (NETBEANS, 2005). A definição dos arquivos XML que dão suporte a geração de interfaces para a avaliação e que armazena os dados das avaliações está sendo realizada através do XML Spy 2005 (ALTOVA, 2005).

Ainda está em fase de desenvolvimento, as operações de criação automática de interfaces para a extração de requisitos não funcionais. Essas operações devem permitir a definição de que eventos devam ser capturados e de como as interfaces devam ser geradas para se conseguir aferir os valores de requisitos não funcionais. Atualmente apenas um conjunto fixo de requisitos não funcionais estão disponíveis em FASE.

No capítulo a seguir, apresentamos algumas conclusões acerca desse trabalho, bem como suas contribuições dentro desse processo tão complexo, que é o de avaliação, e o fruto dessas contribuições através dos trabalhos publicados, juntamente com os comentários sobre os trabalhos futuros.

# 8 Conclusão

Os inúmeros softwares educativos disponíveis atualmente fornecem mecanismos que podem auxiliar de modo efetivo a aprendizagem de conceitos, dos mais simples aos mais elaborados. Contudo, essa disponibilidade de SE tem causado a proliferação de produtos dos mais diversos tipos. Atualmente, podemos encontrar inúmeros SE que trabalham conceitos semelhantes, o que tem dificultado a seleção de um determinado produto de software a ser utilizado em sala de aula.

Neste trabalho, argumentamos que é importante o tratamento preciso de aspectos funcionais e não funcionais como um modo de realizar uma avaliação voltada para a seleção de produtos de SE de mesmo domínio de aplicação. Além de ser necessário a consideração de aspectos pedagógicos/cognitivos nesse processo de avaliação. Dessa forma, apresentamos uma metodologia de suporte a avaliação comparativa de SE. Essa metodologia considera que o próprio avaliador deve escolher que requisitos são importantes para ele e, a partir daí, realizar a definição e especificação de métricas para serem aferidas nos produtos de software que estão sendo comparados.

Na metodologia que propomos, essas métricas são ajustadas para serem utilizadas como insumos para a aplicação da técnica de pontos de casos de uso, amplamente utilizada na realização de estimativas de esforço de desenvolvimento de produtos de software. Através dessa abordagem, conseguimos gerar uma pontuação para cada software, permitindo a realização de uma seleção mais refinada e que considera diversos aspectos relevantes aos SE que vão além da sua interface.

No intuito de automatizar esse processo de avaliação, propomos ainda a criação de uma ferramenta denominada FASE, que tem como intuito facilitar o processo de avaliação para os usuários que não tenham conhecimentos prévios de termos próprios da Teoria dos Campos Conceituais, bem como da própria engenharia de software, e que vejam esses conhecimentos prévios como obstáculos para a utilização da metodologia.

## 8.1 Contribuições e Trabalhos Futuros

A proposta inicial deste trabalho era criar uma metodologia de avaliação de software educativo que fosse mais abrangente, no sentido de tornar possível a escolha de software educativo de mesma área de aplicação por parte dos professores. No entanto, com o desenvolvimento deste trabalho verificamos que era possível criarmos não somente uma metodologia, mas também uma ferramenta que iria traduzir toda a metodologia que propomos, de forma a torná-la realmente acessível a todos aqueles que se importam em avaliar a qualidade do produto que será utilizado em sala de aula. Diante da conclusão deste trabalho, descrevemos a seguir as nossas contribuições.

Procuramos definir uma estratégia a ser utilizada na avaliação comparativa de software educativo, em que o objetivo final era facilitar a seleção de produtos de software educativo, bem como a criação de uma metodologia de suporte a essa nova forma de avaliação.

Na nossa abordagem realizamos a aplicação da técnica de Pontos de Casos de Uso, amplamente usada em engenharia de software para dimensionar o custo do desenvolvimento de software, como um suporte a avaliação de software, conciliada a utilização da Teoria dos Campos Conceituais, como forma de capturar os fatores pedagógicos/cognitivos.

Por fim, a proposta de automatização do processo de avaliação, através da criação de uma ferramenta denominada FASE, com o intuito de facilitar a tarefa de avaliação de software educativo.

Apresentamos ainda as nossas pretenções quanto aos trabalhos futuros, são elas:

A expansão da ferramenta FASE, para uma ferramenta que seja capaz de avaliar diversos software ou seja, ao final da avaliação o usuário terá condições de ver todos os resultados de suas avaliações simultaneamente.

Procurar expandir também a metodologia e ferramenta que desenvolvemos para avaliação de software educativo e utiliza-la na avaliação de objetos de aprendizagem.

Adicionalmente, procuraremos fazer uso de outras teorias, que abordem áreas distintas da matemática, para expandir nossa metodologia.

## 8.2 Trabalhos Publicados

Como resultado de nosso trabalho, obtivemos quatro trabalhos aceitos distribuídos entre periódiocos e conferências nacionais e internacionais. A seguir, detalharemos melhor cada

publicação.

Durante o primeiro semestre de 2004, submetemos o nosso primeiro trabalho para o Simpósio Brasileiro em Informática Educativa - Sbie2004. Neste período estávamos amadurecendo a idéia de nosso trabalho. Nesse artigo, propusemos uma nova metodologia para avaliação de software que fazia uso de técnicas de engenharia de software juntamente com a Teoria dos Campos Conceituais. Por ser o início de nosso trabalho, os conceitos relacionados a engenharia de software e a Teoria dos Campos Conceituais com os quais trabalhamos foram superficiais. No segundo semestre de 2004, resolvemos aprofundar esses conceitos, no quesito relacionado a engenharia de software, conseguimos conciliar a técnica de pontos de casos de uso com a Teoria dos Campos Conceituais que acabou resultando em um artigo que foi publicado na Revista Latinoamericana de Tecnologia Educativa - Relatec2004. Dando continuidade a nossa pesquisa, visto que já havíamos definido os critérios que estavamos utilizando relativos à engenharia de software, partimos agora para uma melhor agregação da Teoria dos Campos Conceituais de forma a torná-la tão determinante em nossa metodologia, quanto a técnica de pontos de casos de uso. Dessa forma, desenvolvemos uma forma de correlacionar essa teoria com a nossa metodologia. Para verificar o resultado, submetemos o nosso terceiro artigo para o VI Encuentro Internacional sobre Educación, Capacitación Profesional, Tecnologías de la Información e Innovación Educativa - Virtual Educa 2005, tendo sua aprovação. Ainda no intuito de aprofundar a relação da Teoria dos Campos Conceituais com o nosso trabalho, submetemos o nosso quarto trabalho agora para A 5th IEEE International Conference on Advanced Learning Technologies - ICALT2005, no qual obtivemos aprovação. De posse desses resultados, foi desenvolvida essa proposta.

# Referências Bibliográficas

ALBRECHT, A. J.; GAFFNEY, J. E. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering*, v. 9, n. 6, p. 639–648, 1983.

ALTOVA. XML Spy 2005. 2005. Disponível em: <http://www.altova.com/products_ide.html.>. Acesso em: 29 setembro 2005.

BASTIEN, J. M. C.; SCAPIN, L. D. *Ergonomic Criteria for the Evaluation of Human-Computer Interfaces*. França, 1993.

BATISTA, S. C. F. *Softmat: Um Repositório de Softwares para Matemática do Ensino Médio - Um Instrumento em Prol de Posturas mais Conscientes na Seleção de Softwares Educacionais*. Dissertação (Mestrado) — Universidade Estadual do Norte Fluminense - Programa de Pós-Graduação em Engenharia de Produção, Rio de Janeiro, 2004.

BLAYA, C. Processo de Avaliação. *Núcleo de Estudos, Experiências e Pesquisas em Trabalho, Movimentos Sociais e Educação*, p. 2–30, Julho 2004. Disponível em: <http://www.ufrgs.br/tramse/med/textos/2004_07_20_tex.htm>. Acesso em: 04 agosto 2004.

BOEHM, B. *Software Cost Estimation With COCOMO II*. [S.l.]: Prentice-Hall, 2000.

CASTRO FILHO, J. A. de. Balança Interativa [Software]. 2004. Disponível em: <http://www.vdl.ufc.br/ativa/balanca_interativa.htm>. Acesso em: 21 janeiro 2004.

GAMEZ, L. *Técnica de Inspeção de Conformidade Ergonômica de Software Educacional*. Dissertação (Mestrado) — Universidade do Minho - Programa de Pós-Graduação em Engenharia Humana, Portugal, 1998.

GLADCHEFF, A. P.; ZUFFI, E.; SILVA, M. da. Um Instrumento para Avaliação da Qualidade de Softwares Educacionais de Matemática para o Ensino Fundamental. In: *VII Workshop Sobre Informática na Escola*. Fortaleza: [s.n.], 2001.

GOMES, A. S. et al. Avaliação de Software Educativo para o Ensino de Matemática. In: *VIII Workshop Sobre Informática na Escola*. Florianópolis: [s.n.], 2002.

H.A.LINSTONE; M.TUROFF. *The Delphi Method: Techniques and Applications*. [S.l.]: Addison-Wesley, 1975.

HEEMANN, V. *Avaliação Ergonômica de Interfaces de Bases de Dados por Meio de CHECKLIST Especializado*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina - Programa de Pós-Graduação em Engenharia de Produção, Florianópolis, 1997.

HUMPHREY, W. S. Predicting (Individual) Software Productivity. *IEEE Transactions on Software Engineering*, v. 17, p. 196–207, 1991.

KARNER, G. *Metrics for Objectory*. Tese (Doutorado) — University of Linköping, Sweden, December 1993.

LUCA. *Jogo da Balança. Software: CD-ROM*. 1995.

MAGINA, S. et al. *Repensando a Adição e a Subtração: contribuições da Teoria dos Campos Conceituais*. São Paulo: PROEM-PUC/SP, 2001.

MICROSYSTEMS, S. Java 2 Platform, Standard Edition (J2SE). 2005. Disponível em: <http://java.sun.com/j2se/index.jsp>. Acesso em: 29 setembro 2005.

MOREIRA, M. A. A Teoria dos Campos Conceituais de Vergnaud, o Ensino de Ciências e a Pesquisa nesta Área. *Investigações em Ensino de Ciências*, v. 7, n. 1, Porto Alegre 2002.

NETBEANS. NetBeans 5.0. beta 2. 2005. Disponível em: <http://www.netbeans.org/>. Acesso em: 29 setembro 2005.

NIELSEN, J. Usability Metrics. January 2001. Disponível em: <http://www.useit.com/alertbox/20010121.html>. Acesso em: 29 setembro 2005.

OLIVEIRA, C. A. de; KLUPPEL, Z. E. Softwares Educacionais de Matemática: Avaliá-los? In: *X Simpósio Brasileiro de Informática na Educação - SBIE*. Curitiba: UFPR, 1999.

OLIVEIRA, C. C.; COSTA, J. W.; MOREIRA, M. *Ambientes Informatizados de Aprendizagem: Produção e Avaliação de Software Educativo*. Campinas: Papirus, 2001. 594 p.

OLIVEIRA, S. S.; GOMES, A. S.; BORGES NETO, H. Avaliação de Software Educativo para o Ensino de Matemática - O Caso das Estruturas Aditivas. In: *Encontro de Pesquisa Educacional do Nordeste: Educação, Desenvolvimento Humano e Cidadania*. São Luís: [s.n.], 2001. p. 594.

PAPERT, S. *Logo: Computadores e Educação*. [S.l.]: Brasiliense, 1985.

PINTO, J. Laboratórios de Matemática no Ensino Básico/Secundário. Setembro 2001. Disponível em: <http://www.prof2000.pt/users/j.pinto/materiais/Lab_mat.doc>. Acesso em: 30 setembro 2005.

PRESSMAN, R. *Software Engineering: A Practitioner's Approach*. [S.l.]: McGraw-Hill College, 2004.

PUTNAM, L. H.; MYERS, W. *Measures for excellence: Reliable software on time, within budget*. [S.l.]: Yourdon Press, 1992.

REEVES, T.; HARMON, S. W. *Systematic Evaluation Procedures for Interactive Multimedia for Education and Training*. May 1996.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. *The Unified Modeling Language Reference Manual*. [S.l.]: Addison-Wesley, 1998.

SILVA, C. M. T. da. Avaliação de Software Educacional. *Conecta - Revista on-line de Educação a Distância*, n.4, 2002. Disponível em: <http://www.revistaconecta.com/conectados/christina_avaliacao.htm>. Acesso em: 04 agosto 2004.

SILVA, C. R.; VARGAS, C. L. S. Avaliação da Qualidade de Software Educacional. In: *XIX Encontro Nacional de Engenharia de Produção e V International Congress of Industrial Engineering*. Rio de Janeiro: Anais. CD-ROM, 1999.

SILVA, C. R. O. E. *Bases Pedagógicas e Ergonômicas para Concepção e Avaliação de Produtos Educacionais Informatizados*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina - Programa de Pós-Graduação em Engenharia de Produção, Florianópolis, 1998.

SILVA, C. R. O. E. *MAEP: Um Método Ergopedagógico Interativo de Avaliação para Produtos Educacionais Informatizados*. Tese (Doutorado) — Universidade Federal de Santa Catarina - Programa de Pós-Graduação em Engenharia de Produção, 2002.

SOMMERVILLE, I. *Software Engineering*. 6th. ed. [S.l.]: Addilson Wesley, 2003.

SOUZA, M. F. C. et al. Uma Metodologia de Apoio à Seleção de Softwares Educativos para o Ensino de Matemática. *Revista Latinoamericana de Tecnologia Educativa - RELATEC*, v. 3, p. 61–83, 2004. Disponível em: <http://158.49.119.99/crai/personal/relatec/VOL3_2/Pequeno.pdf>. Acesso em: 29 setembro 2005.

SOUZA, M. F. C. de; CASTRO FILHO, J. A. de; PEQUENO, M. C. Uma Abordagem Semi-Automática para a Avaliação Comparativa de Software Educacional de Matemática. In: *XV Simpósio Brasileiro de Informática na Educação*. Manaus: UFMA, 2004.

SOUZA, M. F. C. de; PEQUENO, M. C.; CASTRO FILHO, J. A. de. A Software Evaluation Approach Based on Vergnaud s Conceptual Fields Theory. In: *The 5th IEEE International Conference on Advanced Learning Technologies*. Taiwan: IEEE, 2005.

SOUZA, M. F. C. de; PEQUENO, M. C.; CASTRO FILHO, J. A. de. Professor x Software Educativo: a Difícil Tarefa de Escolher. In: *VI Encuentro Internacional sobre Educación, Capacitación profesional, Tecnologías de la Información e Innovación Educativa*. Ciudad de México - México: Virtual Educa, 2005.

SYMONS, C. *Software Sizing and Estimating, MKII FPA*. [S.l.]: Jonhn Wiley and Sons, 1991.

TALL, D. *Concept Image and Concept Definition*. [S.l.]: Jan de Lange, Michiel Doorman, 1988. 37-41 p. (Senior Secundary Mathematics Education).

TEODORO, V.; FREITAS, J. C. *Educação e Computadores*. [S.l.]: Min.Edu./GEP, 1992.

VALENTE, J. A. *O Computador na Sociedade do Conhecimento*. São Paulo: NIED, 1999.

VERGNAUD, G. Multiplicative Structures. *Acquisition of Mathematical Concepts and Processes*, p. 127–174, 1983.

VERGNAUD, G. La Théorie des Champs Conceptuels. In: *Recherches en Didactique des Mathématiques*. [S.l.]: La Pensée Sauvage Editeurs, 1990. p. 133–170.