UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

CONFIGURAÇÃO DE SISTEMAS COMPLEXOS

Aluna: Letícia Lacerda Silva
Orientador: Prof. Dr. Fábio Nogueira de Lucena

Goiânia

2003

LETÍCIA LACERDA SILVA

CONFIGURAÇÃO DE SISTEMAS COMPLEXOS

Projeto Final de Curso apresentado ao Curso de Especialização em Análise e Projeto de Sistemas de Informação do Instituto de Informática da Universidade Federal de Goiás, para obtenção do título de Especialista em Análise de Sistemas de Informação.
Área de Concentração: Análise e Desenvolvimento de Software.
Orientador: Prof. Dr. Fábio Nogueira de Lucena

Goiânia

2003

LETÍCIA LACERDA SILVA

CONFIGURAÇÃO DE SISTEMAS COMPLEXOS

Projeto Final de Curso defendido e aprovado em 07 de maio de 2003, pela Banca Examinadora constituída pelos professores.

_____

Professor Dr. Fábio Nogueira de Lucena

Presidente da Banca

_____

Professor Dr. Carlos Alberto M. Barbosa

_____

Professor Ms. Dirson Santos de Campos

Aos meus pais e amigos que me incentivaram durante o desenvolvimento deste curso de pós-graduação.

A Deus por ter me dado forças e por ter me iluminado durante a execução deste trabalho.

AGRADECIMENTOS

Em primeiro lugar, aos professores do curso de especialização, pelo carinho e comprometimento concedidos durante a execução do curso. E, em especial ao professor e orientador dessa monografia, professor Dr. Fábio Nogueira de Lucena, pela enorme contribuição com as idéias, experiência, apoio e o tempo dispensados durante o desenvolvimento deste trabalho.

Agradecimentos especiais também aos familiares que me apoiaram e entenderam a minha ausência.

"Em todas as ocasiões, mantenha a mente aberta para a mudança. Receba-a de braços abertos. Corteja-a. Somente através do exame e reexame de suas opiniões e idéias você poderá evoluir."

DALE CARNEGIE.

SUMÁRIO

LISTA DE FIGURAS ..... **IX**

LISTA DE ABREVIATURAS ..... **X**

RESUMO ..... **XI**

ABSTRACT ..... **XII**

1. INTRODUÇÃO **13**

1.1. OBJETIVOS DO TRABALHO 14
1.2. JUSTIFICATIVA 14
1.3. METODOLOGIA ADOTADA 15
1.4. ORGANIZAÇÃO DO RELATÓRIO 16

2. XML **17**

2.1. CARACTERÍSTICAS 17
2.2. COMPARAÇÕES ENTRE XML E HTML 18

3. REGRAS QUE REGEM O SISPG **20**

3.1. EXEMPLIFICAÇÃO DETALHADA DE ALGUMAS REGRAS 20
3.2. MAPEAMENTO DE TODAS AS REGRAS CONTEMPLADAS NESTE TRABALHO 24
3.3. TABELA DE ACESSO RÁPIDO ÀS REGRAS 33

4. APLICAÇÃO DE MANUTENÇÃO DAS REGRAS **38**

4.1. VISÃO GERAL 38
4.2. REQUISITOS FUNCIONAIS 38
4.2.1. Ler e salvar arquivos XML no disco *39*
4.2.2. Excluir elementos de Regras do arquivo XML *39*
4.2.3. Inserir elementos de Regras no arquivo XML *39*
4.2.4. Alterar elementos de Regras no arquivo XML *39*
4.2.5. Garantir conformidade do arquivo XML com o seu DTD *40*
4.2.6. Gerar um arquivo XML padrão *40*
4.3. REQUISITOS NÃO FUNCIONAIS 40
4.3.1. Fácil manutenção *40*
4.3.2. Facilitar o aprendizado e o uso *40*
4.3.3. Confiabilidade das informações *41*
4.4. DIAGRAMAS DE CASOS DE USO 41
4.4.1. Descrição dos Casos de Uso *42*
4.5. MODELO DE DOMÍNIO 45

5. PROTÓTIPO DA APLICAÇÃO DE MANUTENÇÃO DAS REGRAS **46**

5.1. APRESENTAÇÃO DAS CLASSES QUE COMPÕEM O PROJETO 46
5.2. DETALHAMENTO DAS CLASSES EMPREGADAS 47
5.2.1. Classe AdapterNode *47*
5.2.2. Classe AlterarRegra *48*
5.2.3. Classe DomToTreeModelAdapter *48*
5.2.4. Classe InserirRegra *49*
5.2.5. Classe MenuPrincipal *49*
5.2.5.1. Opções de Menu disponíveis na Classe MenuPrincipal 58
5.2.6. Classe ParserErrorHandler *58*
5.2.7. Classe ValidateXMLAction *59*
5.2.8. Classe XmlFileFilter *60*
5.2.9. Classe XMLUtils *60*
5.3. MANUTENÇÃO NO ARQUIVO XML E NO DTD 60
5.3.1. Passos para alterar o arquivo XML e o DTD *61*

6. INTERFACE GRÁFICA DO SOFTWARE REGRAS..... 63

7. CONCLUSÃO..... 66

ÍNDICE REMISSIVO..... 67

REFERÊNCIAS BIBLIOGRÁFICAS..... 69

LISTA DE FIGURAS

Figura 2.1: Exemplo de informações em HTML e o arquivo XML correspondente 19
Figura 4.1: Diagrama de casos de uso do software Regras 41
Figura 4.2: Representação de exclusão de Regras 44
Figura 4.3: Modelo de Domínio do software Regras 45
Figura 5.1: Representação das classes do software Regras 46
Figura 5.2: Exemplificação de elementos com tags repetidas 54
Figura 5.3: Alteração de Atributos no arquivo DTD (Regra 19) 61
Figura 5.4: Alteração de Regra com Atributo Título 62
Figura 5.5: Alteração de Atributos no arquivo DTD (Regra 02) 62
Figura 6.1: Opções do Menu da Classe MenuPrincipal 63
Figura 6.2: Exemplificação da Classe AlterarRegra 63
Figura 6.3: Exemplificação da Classe InserirRegra 64
Figura 6.4: Alteração de regra com atributos booleanos (Sim ou Não) 64
Figura 6.5: Alteração de regra com atributos dos dias da semana 65
Figura 6.6: Alteração de regra com atributos das unidades 65
Figura 6.7: Alteração de regra que não possui atributos 65

LISTA DE ABREVIATURAS

DTD

Document Type Definitions.

HTML

HiperText Markup Language.

SGML

Standard Generalized Markup Language.

SISPG

Sistema de Gerenciamento da Pesquisa e da Pós-Graduação da UFG.

UFG

Universidade Federal de Goiás.

XML

Extensible Markup Language.

RESUMO

O SISPG – Sistema de Gerenciamento da Pós-Graduação da UFG é um sistema Complexo, composto por vários componentes. Um dos desafios deste sistema é a configuração das regras de negócio da pós-graduação, relativamente voláteis. Por exemplo, durante a realização deste trabalho todo o conjunto de regras foi revisto e várias mudanças foram realizadas.

O presente trabalho visa analisar as regras de negócio que regem o SISPG, a posterior produção de uma proposta para registro delas e um protótipo de aplicação que gerencie o conjunto de regras.

ABSTRACT

SISPG – System of management of posgraduate of the UFG is a complex system, and formed by a lot of components. One of the challenge of this system is the configuration of the business rules of the postgraduate, more or less changeable. For example, during the realization of this work, all the rules were revised and a lot of changes were implemented.

The propose of this work is to analyse the business rules, which conduct the SISPG, after that, we create a prototype to catalogue those rules, and a prototype that will manage them.

1. INTRODUÇÃO

Atualmente, as atividades desenvolvidas na pós-graduação *stricto sensu* e *lacto sensu* da UFG não contam com ferramentas específicas e integradas para facilitar o trabalho dos administradores e da clientela dos respectivos cursos e, conseqüentemente reduzir os erros humanos e aumentar a eficiência dos processos realizados.

As atividades atualmente manuais conduzem à baixa credibilidade das informações, divergência de dados, baixa produtividade, e também, frustração dos interessados. Em conseqüência, encontra-se em andamento o desenvolvimento de um sistema que almeja organizar e gerenciar as informações pertinentes à pós-graduação da UFG, além de garantir a integridade dos dados e maior agilidade na obtenção das informações.

A análise dos requisitos e funcionalidades da realidade da pós-graduação da UFG [Visão2002] mostrou a clara necessidade de desenvolver um sistema específico, pois os softwares atualmente empregados em outros ambientes semelhantes ao da UFG não atendem aos requisitos identificados, em parte, pela significativa quantidade de regras de negócio correspondentes.

Em sistemas complexos, que possuem vários componentes interagindo entre si para alcançar um objetivo comum, é significativa a necessidade de configurar a execução destes componentes e as regras do negócio mantidas por estes.

O presente projeto tem justamente a finalidade de analisar as Regras que regem a pós-graduação da UFG, estudar a volatilidade de seus valores e apresentar uma proposta que automatize a manutenção de tais regras [Regras2002].

Para exemplificar algumas das regras da pós-graduação, podemos citar:

- Regra para aprovação ou reprovação de um aluno.
- Regra de definição da freqüência mínima para cada disciplina.
- Regra de duração do curso.
- Regra para aproveitamento de créditos.

O estudo dos documentos pertinentes ao SISPG permitiu o levantamento das Regras de interesse. Cada regra foi configurada com um valor que representa a realidade atual da pós-graduação da UFG. Porém, para atender as características vigentes em determinado cenário, tais regras poderão ser alteradas. Em muitos casos, a alteração dos valores das regras é um trabalho difícil, pois estão dispersos em vários componentes. Este contexto de volatilidade do valor configurado para as regras, exige um componente responsável por efetuar a manutenção destes valores, tema do presente trabalho.

1.1. Objetivos do Trabalho

Este trabalho tem como objetivos principais, a análise da melhor maneira para representação das regras que regem a pós-graduação da UFG e a confecção de um software, que irá automatizar as etapas de manutenção das informações pertinentes às regras.

1.2. Justificativa

As regras de negócio da pós-graduação apresentam certa volatilidade. Considerar os valores correspondentes como parte de programas do SISPG é uma solução insatisfatória, pois a volatilidade supracitada implicaria em constantes manutenções no software. Esta situação também é a mesma para a configuração de informações próprias dos vários componentes a serem produzidos.

Para amenizar as necessidades de manutenções futuras, as informações correspondentes deveriam se manter “fora” do SISPG, onde possam ser alteradas sem o emprego de programadores. Reduzir as inevitáveis manutenções futuras de software do SISPG é um objetivo desejável e o corrente trabalho colabora com a obtenção deste objetivo.

Convém ressaltar contudo, que o presente trabalho oferece uma solução prática para a manutenção de valores das regras, mas não considera, por exemplo, a eliminação e o acréscimo de regras. A alteração do conjunto de regras exige alteração no software correspondente.

1.3. Metodologia Adotada

Este trabalho desenvolveu duas grandes atividades:

1. Definir como as regras de negócio da pós-graduação podem ser registradas;
2. Desenvolver protótipo que valide o modelo de registro proposto.

Armazenamento das Regras do Negócio

As atuais regras que regem a pós-graduação da UFG foram analisadas, com o objetivo de definir uma forma para que as mesmas pudessem ser adequadamente armazenadas.

Estabeleceu-se que XML seria o meio e as regras de formação do arquivo XML correspondente estariam definidas em um DTD.

Detalhes sobre as tecnologias empregadas podem ser obtidas no capítulo seguinte.

Manutenção das Regras

Através de um programa deverá ser possível consultar todos os conteúdos das regras presentes no arquivo XML, além de permitir inclusões e alterações dos valores das regras já existentes.

Este programa deverá permitir alterações no conteúdo das regras sem que alterações em software, realizadas por programadores sejam necessárias. Convém ressaltar neste ponto, que, nem "tudo" pode ser antecipado ou está suscetível a tal abordagem. Por exemplo, alterar o período máximo de prorrogação de um programa de mestrado poderá ser contemplado facilmente pela presente proposta. Por outro lado, uma regra pode ser acrescida, apenas com o propósito de ilustrar: uma regra que impeça a inscrição em programas de mestrado por estudantes que realizaram a graduação em instituição localizada fora do estado de Goiás. Este caso "extravagante", não tem como ser antecipado e, dessa forma, alterações realizadas por programadores deverão ser efetuadas para contemplar o acréscimo desta regra. Em resumo, o acréscimo de novas regras não será permitido pelo presente programa.

1.4. Organização do Relatório

Este documento está organizado da seguinte forma:

O Capítulo 1 introduz a situação atual da pós-graduação da UFG e das Regras que a regem, além de apresentar os objetivos, a justificativa do trabalho e a metodologia de trabalho adotada.

O Capítulo 2 apresenta ao leitor as características da linguagem XML, empregada no desenvolvimento do software proposto e um comparativo entre as linguagens XML e HTML.

No Capítulo 3 estão esboçadas as regras que regem a pós-graduação da UFG, algumas com explicações detalhadas e outras com menor número de detalhes. No final deste capítulo também encontramos uma tabela de acesso rápido às Regras.

O Capítulo 4 detalha todos os requisitos funcionais e não funcionais eliciados, bem como o Diagrama de Casos de Uso e o Modelo de Domínio do software Regras.

O Capítulo 5 contém a descrição detalhada de todas as classes de projeto, onde são descritos seus objetivos e todos os seus métodos. Este capítulo contém todas as instruções necessárias para alguma possível manutenção do software Regras.

O Capítulo 6 mostra ao leitor a interface gráfica do software Regras, com explicações de sua utilização.

O Capítulo 7 traz as conclusões a respeito do trabalho e as vantagens de sua utilização dentro do contexto do SISPG.

O relatório apresenta ainda um Glossário dos termos utilizados, um índice remissivo dos termos mais empregados e as referências bibliográficas utilizadas durante a confecção deste trabalho.

2. XML

2.1. Características

Extensible Markup Language (XML) é uma linguagem de marcação de dados (meta-markup language) que provê um formato para descrever dados estruturados. Isso facilita declarações mais precisas do conteúdo e resultados mais significativos de busca através de múltiplas plataformas [XML2003].

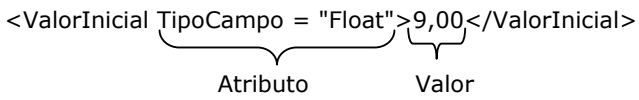
XML é um subconjunto do SGML (Standard Generalized Markup Language), o qual é otimizado para distribuição através da Web, e é definido pelo Word Wide Web Consortium (W3C), assegurando que os dados estruturados serão uniformes e independentes de aplicações e fornecedores.

XML apresenta uma série de vantagens, conforme citado abaixo:

- Fornece uma representação estruturada dos dados que mostrou ser amplamente implementável e fácil de ser desenvolvida.
- Provê um padrão que pode codificar o conteúdo, as semânticas e as esquematizações para uma grande variedade de aplicações, desde as mais simples até as mais complexas, dentre elas: um simples documento; um registro estruturado tal como uma ordem de compra de produtos; um objeto com métodos e dados como objetos Java ou controles ActiveX; apresentação gráfica, como interface de aplicações de usuário; entidades e tipos de esquema padrões; todos os links entre informações e pessoas na Web.
- XML também é considerada de grande importância na Internet e em grandes Intranets porque provê a capacidade de interoperação dos computadores, por ter um padrão flexível, aberto e independente de dispositivo. As aplicações podem ser construídas e atualizadas mais rapidamente e também permitem múltiplas formas de visualização dos dados estruturados.

Em XML as regras que definem um documento são ditadas por DTDs (Document Type Definitions), as quais ajudam a validar os dados, quando a aplicação que os recebe não possui internamente uma descrição do dado que está recebendo. Um analisador de documentos pode checar os dados do arquivo XML através da análise das regras contidas no DTD, para ter certeza de que o dado foi estruturado corretamente.

Uma tag em XML pode conter atributos ou valores. Atributo é uma característica da tag localizada dentro dos seus sinais delimitadores (‘<’ e ‘>’). Já o valor da tag está localizado entre as tags de início e fim da informação que está sendo mapeada. Observe abaixo:



O encerramento de uma tag que possui valor é diferente do encerramento da tag que não possui valor configurado. Para as tags que possuem valor configurado, é necessário criar uma tag com o nome original da tag, acrescida do sinal ‘/’, após o valor configurado. Conforme mostrado abaixo:

<ValorInicial TipoCampo = "Float"> 9,00</ValorInicial>

Já para as tags que não possuem valor configurado, basta inserir o sinal ‘/’ antes do sinal ‘>’ da tag. Observe o exemplo:

<ExameMestrado Obrigatório = "S" Regra="12"/ >

2.2. Comparações entre XML e HTML

XML e HTML são muito parecidas. Ambas identificam elementos em uma página e ambas utilizam sintaxes similares.

A mais importante característica da linguagem XML se resume em separar a interface com o usuário (apresentação) dos dados estruturados. HTML especifica como os dados estruturados são apresentados. Por exemplo, em HTML são utilizadas tags para definir tamanho e cor de fonte de informações sobre referências. Já em XML você utiliza as tags para descrever apenas os dados, como exemplo tags de assunto, título, etc...

Tal qual HTML, XML também faz uso de tags (palavras encapsuladas por sinais '<' e '>') e atributos (definidos com name = "value"), mas enquanto HTML especifica a apresentação das tags e dos atributos, XML usa as tags somente para delimitar trechos de dados, e não inclui a interpretação visual.

Por exemplo, enquanto em um documento HTML uma tag <p> indica um parágrafo, em XML essa tag pode indicar um preço, um parâmetro, uma pessoa, ou

qualquer outra coisa que se possa imaginar (inclusive algo que não tenha nada a ver com um “p” como, por exemplo, autores de livros).

Exemplo de estruturas HTML e XML

Abaixo, iremos comparar a implementação em XML e HTML da mesma tabela exemplificadora de configuração de regras: [WEB APPLICATIONS]

Observe que, ao contrário do conteúdo do arquivo em XML, aquele em HTML possui “interpretação visual” bem-definida.

Data	99/99/9999
Nota Mínima	7,0
Freqüência Mínima	85%

- HTML

```
<html>
    <body>
        <table>
            <tr> <td> Data </td> <td> 99/99/9999 </td> </tr>
            <tr> <td> Nota Mínima </td> <td> 7,0 </td> </tr>
            <tr> <td> Freqüência Mínima </td> <td> 85% </td> </tr>
        </table>
    </body>
</html>
```

- XML

```
<regras>
    <data> 99/99/9999 </data>
    <Nota Mínima> 7,0 </Nota Mínima>
    < Freqüência Mínima> 85 </ Freqüência Mínima>
</regras>
```

Figura 2.1: Exemplo de informações em HTML e o arquivo XML correspondente

3. Regras que regem o SISPG

Neste capítulo serão apresentadas as regras que regem a pós-graduação da UFG e como elas podem ser registradas via emprego de XML.

A seção 3.1 documenta extensivamente como este registro é possível. Nesta seção apenas algumas regras são consideradas. Já a seção 3.2 apresenta todas as regras contempladas no arquivo XML, com um número menor de detalhes, o que não comprometerá o entendimento por parte dos leitores, dada a semelhança com as regras contempladas nesta primeira seção.

3.1. Exemplificação detalhada de algumas Regras

Conceito de avaliação nas Disciplinas (Regra 2)

Na UFG, o conceito para aprovação ou reprovação em alguma disciplina é um valor no intervalo de 0 a 10, inclusive, com a precisão de duas casas decimais. Alternativamente, pode-se "mapear" tais valores em intervalos de A a F.

Suponha que o mapeamento das Notas no ano vigente obedeça à seguinte regra: o conceito A agrupa as notas de 9,00 até 10,00; o conceito B agrupa as notas de 8,00 até 8,99; o conceito C as notas de 7,00 até 7,99; o conceito D as notas de 6,00 até 6,99; o conceito E as notas de 5,00 até 5,99 e o conceito F as notas de 0,00 até 4,99. Agora, imagine que este conceito deverá ser alterado para o próximo ano letivo, e o conceito A passará a representar as notas de 8,50 até 10,00 e assim por diante nos demais conceitos. A volatilidade dos valores configurados para os conceitos das notas da UFG implicou na sua configuração no arquivo XML, onde será necessário armazenar o conceito (uma letra que o represente) e o intervalo inicial e final de notas que ela representa.

Primeiramente, criamos a tag `Conceito`, e dentro dela, a tag `Nota`. O elemento `Nota` poderá ser repetido quantas vezes seja necessário, para que represente o cenário das notas da UFG. Cada tag `Nota` contém as tags `ValorInicial` e `ValorFinal` que representam o intervalo de valores da nota.

Cada tag pode possuir seus próprios atributos. Na representação das regras da pós-graduação da UFG, padronizamos que as tags terão como atributo, o número representativo da sua regra, contido em [Regras2002] e o tipo de campo que pode armazenar (exemplo: string, float, integer, time), além de seus atributos específicos.

Na tag Nota definimos dois atributos. O 1º representa o número da regra, e o 2º representa o conceito atribuído ao intervalo de Nota especificado.

Nas tags ValorInicial e ValorFinal foi definido o atributo TipoCampo que especifica o tipo do campo que a tag pode armazenar.

Para melhor esclarecer estas explicações, segue abaixo o mapeamento da regra Avaliação em Disciplina em XML:

```
<Conceito>
        <Nota Regra = "2" Valor = "A">
                <ValorInicial TipoCampo = "Float">9,00</ValorInicial>
                <ValorFinal TipoCampo = "Float">10,00</ValorFinal>
        </Nota>
        <Nota Regra="2" Valor = "B">
                <ValorInicial TipoCampo = "Float">8,00</ValorInicial>
                <ValorFinal TipoCampo = "Float">8,99</ValorFinal>
        </Nota>
</Conceito>
```

A tag Conceito armazena todas as possíveis variações de intervalos de notas. Por exemplo, a primeira tag de nota representa o conceito A, e inclui as notas no intervalo de 9 a 10 inclusive.

Freqüência mínima para aprovação nas Disciplinas (Regra 4)

Atualmente, a freqüência mínima em cada disciplina da pós-graduação da UFG é fixada em 85%, para todas as disciplinas de todos os programas. Mas, nada impede que no futuro, o valor da freqüência mínima seja alterado. Precisamos permitir que este valor da regra possa ser alterado, para representar o cenário atual da pós-graduação da UFG.

A melhor maneira encontrada para representar as regras da pós-graduação da UFG foi agrupá-las de acordo com sua semântica. Por isso, criamos a tag Tempo, dentro desta, a tag Disciplina, e depois a tag Frequência, que contém os seguintes atributos: Regra

(representa o número da Regra), TipoCampo (armazena o tipo do campo que pode ser armazenado na regra) e Unidade (representa a unidade do valor configurado para a regra).

Pelo conjunto de tags abaixo percebemos que a regra número 4 esclarece a restrição para aprovação em uma determinada disciplina dos cursos de pós-graduação da UFG, e que o valor atualmente configurado é de 85%, além de restringir que o valor armazenado só aceita campos inteiros.

```
<Tempo>
      <Disciplina>
          <Frequência Regra = "4" Unidade = "%" TipoCampo = "Integer"> 85 </Frequência>
      </Disciplina>
</Tempo>
```

Cancelamento de Disciplina (Regra 6)

O cancelamento de disciplina não poderá ocorrer se mais de 15% da disciplina já foi ministrado.

O mapeamento desta regra para XML seguiu o padrão definido e explicado na regra anterior.

As tags abaixo representam o armazenamento da regra de tempo máximo para cancelamento de alguma disciplina no curso de pós-graduação da UFG.

```
<Tempo>
      <Disciplina>
           <Cancelar Regra = "6" Unidade="%" TipoCampo = "Integer">15</Cancelar>
      </Disciplina>
</Tempo>
```

Observe que as duas regras citadas acima tratam do mesmo assunto – Disciplina.

Durante a fase de análise da regras optamos em utilizar a semântica das regras para fazer o seu agrupamento em tags genéricas, com o objetivo de facilitar o entendimento.

Podemos agora, mostrar todo o conteúdo da tag Disciplina.

```
<Tempo>
      <Disciplina>
            <Inscrever Regra="8" Unidade="dia" TipoCampo="Integer">1</Inscrever>
            <Cancelar Regra="6" Unidade="%" TipoCampo="Integer">15</Cancelar>
            <RequisitarAproveitamento Regra="1" Unidade="dia" TipoCampo="Integer">
            2 </RequisitarAproveitamento>
            <Frequência Regra="4" Unidade="%" TipoCampo="Integer">85</Frequência>
            <Categoria>
                  <NomeDisciplina Regra="7" Status="optativa" TipoCampo="String">
                  Análise </NomeDisciplina>
                  <NomeDisciplina Regra="7" Status="obrigatória" TipoCampo="String">
                  Banco de Dados </NomeDisciplina>
            </Categoria>
            <Aproveitamento Regra="16" Unidade="%" TipoCampo="Integer"> 25
            </Aproveitamento>
      </Disciplina>
</Tempo>
```

Estágio docência nas Disciplinas (Regra 10)

Atualmente, o cenário da pós-graduação da UFG determina que o estágio docência é uma atividade de 30 horas (2 créditos) para o mestrado e de 60 horas (4 créditos) para alunos do doutorado. Deverá ser realizado em um período de, no máximo, um semestre para o mestrado e dois semestres para o doutorado.

Mapeamos esta regra no nosso arquivo XML para permitir que o seu valor possa ser alterado toda vez que o cenário exigir.

A representação abaixo exemplifica o mapeamento da regra de Estágio Docência para o Mestrado e Doutorado.

```
<Tempo>
      <EstágioDocencia>
            <EstágioMestrado>
               <HorasMest Regra = "10" TipoCampo = "Integer">30</HorasMest>
               <SemestreMest Regra = "10" TipoCampo = "Integer">1</SemestreMest>
            </EstágioMestrado>
            <EstágioDoutorado>
               <HorasDout Regra = "10" TipoCampo = "Integer">60</HorasDout>
               <SemestreDout Regra = "10" TipoCampo = "Integer">2</SemestreDout>
            </EstágioDoutorado>
      </ EstágioDocencia>
</Tempo>
```

Proficiência em língua estrangeira (Regra 18)

Atualmente, o programa da pós-graduação da UFG exige proficiência em uma língua para o mestrado e em duas línguas para o doutorado.

A volatilidade do valor desta regra exigiu que a incluíssemos no mapeamento do arquivo XML.

Criamos a tag Restrição, que contém a tag LínguaEstrangeira. Dentro da tag LínguaEstrangeira existem outras duas tags: LínguaMestrado e LínguaDoutorado, ambas armazenam um campo do tipo inteiro, e possuem dois atributos: o número da Regra e o tipo do campo que pode ser armazenado na tag.

Abaixo, a sua representação.

```
<Restrição>
      <LínguaEstrangeira>
            <LínguaMestrado Regra = "18" TipoCampo = "Integer">1</LínguaMestrado>
            <LínguaDoutorado Regra = "18" TipoCampo = "Integer">2</LínguaDoutorado>
      </LínguaEstrangeira>
</Restrição>
```

Aqui foram apresentadas algumas das regras da pós-graduação da UFG e as respectivas propostas de registro dos elementos voláteis em XML.

O trabalho de levantamento das regras exigiu a análise da documentação pertinente ao SISPG [Visão2002] [Regras2002] [Casos de Uso] e o estudo da volatilidade de cada regra. Assim, toda restrição de dado, valor, tempo, dentre outras foram mapeadas no arquivo XML.

A seção seguinte exibe o mapeamento obtido. Ao contrário dos exemplos anteriores, os comentários são mínimos, onde a clareza não é comprometida.

3.2. Mapeamento de todas as Regras contempladas neste trabalho

As tags nesta seção estão organizadas de acordo com sua semântica.

A primeira tag do arquivo XML contém as informações de data e hora de última alteração nas regras armazenadas no arquivo XML.

1 – Elementos que envolvem o conceito de Tempo

Restrição de Inscrição em Disciplina (Regra 8)

Explicação: Não pode haver inscrição em disciplina que se encerre após a defesa prevista do aluno em questão.

Mapeamento em XML:

```
<Inscrever Regra = "8" Unidade = "dia" TipoCampo = "Integer">1</Inscrever>
```

Cancelamento de Disciplina (Regra 6 e 24)

Explicação: O cancelamento de disciplina não poderá ocorrer se mais de 15% da disciplina já foi ministrado.

Mapeamento em XML:

```
<Cancelar Regra = "6" Unidade = "%" TipoCampo = "Integer">15</Cancelar>
```

Aproveitamento de Disciplina (Regra 1)

Explicação: O aproveitamento pode ser requisitado até um dia anterior àquele da defesa do aluno.

Mapeamento em XML:

```
<RequisitarAproveitamento Regra = "1" Unidade = "dia" TipoCampo = "Integer"> 1
</RequisitarAproveitamento>
```

Freqüência Mínima em Disciplina (Regra 4)

Explicação: A freqüência mínima em cada disciplina da pós-graduação é fixada em 85% para todas as disciplinas de todos os programas.

Mapeamento em XML:

```
<Frequência Regra = "4" Unidade = "%" TipoCampo = "Integer">85</Frequência>
```

Categorização das Disciplinas (Regra 7)

Explicação: Disciplinas estão organizadas em categorias: obrigatórias ou optativas.

Mapeamento em XML:

```
<Categoria>
      <NomeDisciplina Regra = "7" Status = "optativa" TipoCampo = "String"> Análise
      </NomeDisciplina>
      <NomeDisciplina Regra = "7" Status = "obrigatória" TipoCampo = "String"> Banco de
      Dados </NomeDisciplina>
</Categoria>
```

Aproveitamento de Créditos em Disciplinas (Regra 16)

Explicação: Um total não superior a 25% dos créditos poderá ser aproveitado de disciplinas cursadas em outros programas.

Mapeamento em XML:

```
<Aproveitamento Regra= "16" Unidade="%" TipoCampo= "Integer">25 </Aproveitamento>
```

Estágio Docência – Disciplina (Regra 10)

Explicação: O estágio docência é uma atividade de 30 horas (2 créditos) para o mestrado e de 60 horas (4 créditos) para alunos do doutorado. Deverá ser realizado em um período de, no máximo, um semestre para o mestrado e dois semestres para o doutorado.

Mapeamento em XML:

```
<EstágioMestrado>
      <HorasMest Regra = "10" TipoCampo = "Integer">30</HorasMest>
      <SemestreMest Regra = "10" TipoCampo = "Integer">1</SemestreMest>
</EstágioMestrado>
<EstágioDoutorado>
      <HorasDout Regra = "10" TipoCampo = "Integer">60</HorasDout>
      <SemestreDout Regra = "10" TipoCampo = "Integer">2</SemestreDout>
</EstágioDoutorado>
```

Programa de Estágio Docência (Regra 23)

Explicação: Se o aluno em questão é bolsista da CAPES ou UFG, então o estágio docência é obrigatório.

Mapeamento em XML:

```
<Obrigatório>
      <Instituição Unidade = "S" Regra = "23">UFG</Instituição>
      <Instituição Unidade = "N" Regra = "23">CAPES</Instituição>
</Obrigatório>
```

Trancar Matrícula (Regra 25)

Explicação: O trancamento em matrícula é possível uma única vez por ingresso em programa.

Mapeamento em XML:

```
<Trancar Máximo = "1" Regra = "25"/>
```

Matrícula em Programa (Regra 26)

Explicação: Todo aluno é obrigado a matricular-se periodicamente conforme o regime do programa correspondente.

Mapeamento em XML:

```
<Matricular Regra = "26" Unidade = "dia" TipoCampo = "Integer">2</Matricular>
```

Duração dos Programas de Mestrado e Doutorado (Regra 13)

Explicação: A duração do Mestrado é de no mínimo 18 meses e no máximo de 30 meses. Já no Doutorado a duração mínima é de 24 meses e a máxima de 48 meses.

Mapeamento em XML:

```
<ProgMestrado>
      <TempoMinMest Regra = "13" Unidade = "mês" TipoCampo = "Integer"> 18
      </TempoMinMest>
      <TempoMaxMest Regra = "13" Unidade = "mês" TipoCampo = "Integer">30
      </TempoMaxMest>
</ProgMestrado>
<ProgDoutorado>
      <TempoMinDout Regra = "13" Unidade = "mês" TipoCampo = "Integer"> 24
      </TempoMinDout>
      <TempoMaxDout Regra = "13" Unidade = "mês" TipoCampo = "Integer"> 48
      </TempoMaxDout>
</ProgDoutorado>
```

Prorrogação do prazo de conclusão em programa (Regra 14)

Explicação: Qualquer que seja o nível do programa de um dado aluno, a prorrogação não deverá ser superior a 6 meses.

Mapeamento em XML:

```
<ProrrogarCurso Regra= "14" Unidade="mês" TipoCampo= "Integer"> 6 </ProrrogarCurso>
```

Período máximo de resposta de requerimento (Regra 20)

Explicação: Um requerimento terá 15 dias para a decisão correspondente. Este período será iniciado rigorosamente após o cadastro do mesmo no sistema.

Mapeamento em XML:

```
<ResponderRequerimento Regra = "20" Unidade = "dia" TipoCampo = "Integer"> 15
</ResponderRequerimento>
```

Prazo máximo para apresentar correções na Defesa (Regra 34)

Explicação: Após a defesa de dissertação ou tese, sugestões podem ter sido emitidas e as correções correspondentes, neste caso, terão um prazo limite para serem apresentadas. Atualmente o prazo limite é de 30 dias.

Mapeamento em XML:

```
<CorrigirDissertação Regra = "34" Unidade = "dia" TipoCampo = "Integer">30
</CorrigirDissertação>
```

2 – Elementos que envolvem o conceito de Avaliação

Avaliação em Disciplina (Regra 2)

Explicação: Conceito é um valor no intervalo de 0 a 10, inclusive. Alternativamente, pode-se "mapear" tais valores em intervalos de A a F, por exemplo. Deve-se poder configurar este mapeamento. Também deve ser estabelecido o conceito mínimo para aproveitamento da disciplina.

Mapeamento em XML:

```
<Nota Regra = "2" Valor = "A">
      <ValorInicial TipoCampo = "Float">9,00</ValorInicial>
      <ValorFinal TipoCampo = "Float">10,00</ValorFinal>
</Nota>
<Nota Regra = "2" Valor = "B">
      <ValorInicial TipoCampo = "Float">8,00</ValorInicial>
      <ValorFinal TipoCampo = "Float">8,99</ValorFinal>
</Nota>
```

Resultado Defesa (Regra 33)

Explicação: O resultado da defesa de tese ou dissertação pode ser: (a) aprovado; (b) reprovado e (c) aprovado com pendências. A generalização pode vir de uma lista de possíveis resultados. Cada um deles pode significar: aprovado, reprovado ou pendente.

Mapeamento em XML:

```
<ResultadoDefesa Regra = "33">
        <Opção TipoCampo = "String">Aprovado</Opção>
        <Opção TipoCampo = "String">Pendente</Opção>
        <Opção TipoCampo = "String">Reprovado</Opção>
</ResultadoDefesa>
```

3 – Elementos que envolvem o conceito de Restrição

Exame de Qualificação – obrigatoriedade (Regra 12)

Explicação: Para o doutorado o exame de qualificação é obrigatório. No nível de mestrado, cada programa poderá estabelecer a sua própria política. Ou seja, pode ser obrigatório ou, no outro caso, não existir.

Mapeamento em XML:

```
<ExameQualif>
        <ExameMestrado Obrigatório = "S" Regra="12"/>
        <ExameDoutorado Obrigatório = "N" Regra="12"/>
</ExameQualif>
```

Lista das línguas disponíveis (Regra S[1]1)

Explicação: Há uma lista das línguas aprovadas pela pós-graduação da UFG, como sendo as línguas disponíveis para a escolha das línguas em que o aluno é proficiente.

Mapeamento em XML:

```
<LínguasDisponíveis>
        <Língua Regra="S1" TipoCampo = "String">Inglês</Língua>
        <Língua Regra="S1" TipoCampo = "String">Francês</Língua>
        <Língua Regra="S1" TipoCampo = "String">Português</Língua>
</LínguasDisponíveis>
```

[1] Suplementar

Programa – proficiência em língua estrangeira (Regra 18)

Explicação: É obrigatório para o aluno do mestrado que ele seja proficiente em uma língua; e para o aluno do doutorado, duas línguas.

Mapeamento em XML:

```
<LínguaEstrangeira>
      <LínguaMestrado Regra = "18" TipoCampo = "Integer">1</LínguaMestrado>
      <LínguaDoutorado Regra = "18" TipoCampo = "Integer"> 2 </LínguaDoutorado>
</LínguaEstrangeira>
```

Participações simultâneas dos docentes em Programas (Regra 22)

Explicação: Todo participante de um dado programa também poderá ser participante em, no máximo, outro programa de pós-graduação.

Mapeamento em XML:

```
<Professor>
      <ProgramasSimultâneos Máximo = "2" Regra = "22"/>
</Professor>
```

Matrículas simultâneas por Aluno (Regra 27)

Explicação: Todo aluno está, necessariamente, matriculado em um único programa em um dado instante de tempo.

Mapeamento em XML:

```
<Aluno>
      <MatrículaSimultânea Máximo = "1" Regra = "27"/>
</Aluno>
```

Disponibilidade do SISPG (Regra S2)

Explicação: O SISPG deverá estar apto para operação todos os dias, das segundas-feiras às sextas-feiras, das 07:30 às 17:30.

Mapeamento em XML:

```
<Disponibilidade>
      <SemanaInício Quando = "segunda-feira" Regra="S2"/>
      <SemanaFim Quando = "sexta-feira Regra="S2""/>
      <HoraInício Regra="S2" TipoCampo = "Time">07:30:00</HoraInicio>
```

```
    <HoraFim Regra="S2" TipoCampo = "Time">17:30:00</HoraFim>
</Disponibilidade>
```

Número Máximo de acessos simultâneos ao SISPG (Regra S3)

Explicação: O número máximo de acessos simultâneos ao SISPG é 100. Usuários que requisitarem serviço além deste número serão informados de que o sistema não está apto a atender tantas requisições simultâneas, e solicitará que o usuário tente mais tarde.

Mapeamento em XML:

```
<MaxUsuários Regra="S3" TipoCampo = "Integer">100</MaxUsuários>
```

Programa – Titulação do Corpo Docente (Regra 19)

Explicação: Exceto "outro participante", todas as demais categorias exigem a titulação mínima de doutor. As categorias de participante são: docente, pesquisador e outro participante.

Mapeamento em XML:

```
<Participante>
    <Cargo Regra = "19" Titulo = "doutor" TipoCampo = "String"> Docente </Cargo>
    <Cargo Regra = "19" Titulo = "doutor" TipoCampo = "String"> Pesquisador </Cargo>
    <Cargo Regra = "19" Titulo = "nenhum" TipoCampo = "String">Outro participante
    </Cargo>
</Participante>
```

Solicitação de Requerimentos (Regra 31)

Explicação: Alguns tipos de requerimentos exigem a autorização do orientador. Deve-se mapear todos os requerimentos, e indicar se eles exigem ou não autorização do orientador.

Mapeamento em XML:

```
<SolicitarRequerimento>
    <TipoRequerimento Autorização = "S" Regra = "31" TipoCampo = "String">
    Cancelamento de Matrícula </TipoRequerimento>
    <TipoRequerimento Autorização = "N" Regra = "31" TipoCampo = "String">
    Histórico  Escolar</TipoRequerimento>
</SolicitarRequerimento>
```

4 – Elementos que envolvem o conceito de Acessibilidade ao SISPG

Expiração de sessão no SISPG (Regra S4)

Explicação: O tempo máximo de inoperabilidade do SISPG será de 10 minutos, após este tempo a sessão automaticamente expirará.

Mapeamento em XML:

```
<SessãoExpira Regra="S4" Unidade = "min" TipoCampo = "Integer">10</SessãoExpira>
```

Máximo de acessos por usuário no SISPG (Regra S5)

Explicação: Caso um usuário já esteja conectado ao SISPG, e tentar outra conexão, sem antes finalizar a primeira, o sistema irá negar a conexão e enviará uma mensagem para o usuário.

Mapeamento em XML:

```
<Usuário TipoCampo = "Integer">
      <MaxAcesso Regra="S5">1</MaxAcesso>
</Usuário>
```

3.3. Tabela de Acesso Rápido às Regras

Tag	Atributos	Valor	Descrição sucinta
<Regras>			Tag que agrupa todas as Regras que regem a pós-graduação da UFG.
<ÚltimaAlteração>		Data e Hora	Contém a data e hora da última alteração no arquivo XML. Esta tag é automaticamente atualizada quando o usuário altera o valor ou o atributo de alguma regra. A data é a do Sistema Operacional.
<Tempo>			Agrupamento das Regras referentes à Tempo.
<Disciplina>			Agrupamento das Regras referentes à Disciplina.
<Inscrever>	Regra = "8" TipoCampo ="Integer" Unidade = "mês"	1	Um aluno não pode se inscrever em disciplina após a sua defesa.
<Cancelar>	Regra ="6" TipoCampo= “Integer" Unidade = "%"	15	O cancelamento de disciplina não é válido se mais de 15% da aula já foi ministrada.
<Requisitar Aproveitamento>	Regra = "1" TipoCampo= "Integer” Unidade = "dia"	2	Um aluno pode requisitar aproveitamento de disciplina até um dia antes de sua defesa.
<Frequência>	Regra = "4" TipoCampo= “Integer" Unidade = "%"	85	Um dos pré-requisitos para a aprovação dos alunos é a freqüência mínima de 85% das aulas.
<Categoria>			Contém os conceitos de Categoria das Disciplinas.
<NomeDisciplina>	Regra = “7” TipoCampo = “String” Status="obrigatória/ optativa"	Contém os nomes das Disciplinas	
<Aproveitamento>	Regra = “16” TipoCampo= “Integer" Unidade = "%"	25	Somente 25% dos créditos pode ser aproveitado de disciplinas cursadas em outros programas.
<EstágioDocencia>			Agrupamento das Regras referentes à atividade de Estágio Docência.
<EstágioMestrado>			Tag referente à atividade de Estágio Docência no Mestrado.

<HorasMest>	Regra = “10” TipoCampo= “Integer"	30	O estágio docência é uma atividade de 30 horas para o Mestrado.
<SemestreMest>	Regra = “10” TipoCampo= “Integer"	1	O estágio docência deve ser realizado em um semestre no Mestrado.
<EstágioDoutorado>			Tag referente à atividade de Estágio Docência no Doutorado.
<HorasDout>	Regra = “10” TipoCampo= “Integer"	60	O estágio docência é uma atividade de 60 horas para o Doutorado.
<SemestreDout>	Regra = “10” TipoCampo= “Integer"	2	O estágio docência deve ser realizado em dois semestres no Doutorado.
<Obrigatório>			Representa os tipos de bolsas estudantis que exigem a atividade de estágio docência.
<Instituição>	Regra = “23” Unidade = “S/N"		Tag mostra o nome da Instituição que fornece a bolsa, e seu atributo *Unidade* representa se é necessário a atividade de estágio docência.
<Programa>			Agrupamento das regras referentes à Programa.
<Trancar>	Regra = “25” Máximo = “1”		Só é possível trancar uma única vez o programa.
<Matricular>	Regra = “26” TipoCampo= “Integer" Unidade = “dia”	2	Os alunos são obrigados a matricular-se periodicamente conforme o regime do programa.
<ProgMestrado>			Regras referentes ao tempo de término do Mestrado.
<TempoMinMest>	Regra = “13” TipoCampo= “Integer" Unidade = “mês”	18	O tempo mínimo de duração do Mestrado é de 18 meses.
<TempoMaxMest>	Regra = “13” TipoCampo= “Integer" Unidade = “mês”	30	O tempo máximo de duração do Mestrado é de 30 meses.
<ProgDoutorado>			Regras referentes ao tempo de término do Doutorado.
<TempoMinDout>	Regra = “13” TipoCampo= “Integer" Unidade = “mês”	24	O tempo mínimo de duração do Doutorado é de 24 meses.
<TempoMaxDout>	Regra = “13” TipoCampo= “Integer"	48	O tempo máximo de duração do Doutorado é de 48 meses.

<ProrrogarCurso>	Regra = “14” TipoCampo= “Integer" Unidade = “mês”	6	Não é possível prorrogar o prazo de conclusão do curso por um período superior a 6 meses.
<Responder Requerimento>	Regra = “20” TipoCampo= “Integer" Unidade = “dia”	15	O prazo máximo para responder aos requerimentos é de 15 dias.
<CorrigirDissertação>	Regra = “34” TipoCampo= “Integer" Unidade = “dia”	30	O tempo máximo para apresentar as correções na tese de defesa, sugeridas pela banca é de 30 dias.
<Conceito>			Agrupamento das Regras referentes à Conceito.
<ResultadoDefesa>	Regra = “33”		Agrupa os resultados da defesa de tese ou dissertação.
<Opção>	TipoCampo = “String”	Aprovado, pendente, reprovado	Lista os possíveis resultados da defesa de tese ou dissertação.
<Nota>	Regra = “2” Valor = “A”		Representa o mapeamento entre os conceitos de Notas e seu respectivo valor. Esta tag irá se repetir para quantos tipos de conceitos existam.
<ValorInicial>	TipoCampo = "Float"	9,00	Representa o valor inicial da nota cujo conceito está especificado no atributo da sua tag ancestral.
<ValorFinal>	TipoCampo = "Float"	10,00	Representa o valor final da nota cujo conceito está especificado no atributo da sua tag ancestral.
<Restrição>			Agrupamento das regras que envolvem o conceito de Restrição.
<ExameQualif>			Tags referentes ao Exame de Qualificação.
<ExameMestrado>	Regra = “12” Obrigatório = "N"		O exame de qualificação no Mestrado é opcional, depende do programa.
<ExameDoutorado>	Regra = “12” Obrigatório = "S"		O exame de qualificação no Doutorado é obrigatório.
<LínguasDisponíveis>			Agrupa as línguas estrangeiras aceitas na UFG.
<Língua>	Regra = "S1" TipoCampo = "String"	Inglês, Francês, etc	Contém o nome da língua estrangeira. Esta tag será repetida para quantas línguas sejam aceitas na UFG.
<LínguaEstrangeira>			Especifica a proficiência em língua estrangeira.

<LínguaMestrado>	Regra = “18” TipoCampo= “Integer”	1	É necessário ser proficiente em 01 língua no Mestrado.
<LínguaDoutorado>	Regra = “18” TipoCampo= “Integer”	2	É necessário ser proficiente em 02 línguas no Doutorado.
<Professor>			Tag referente aos Professores da pós-graduação da UFG.
<Programas Simultâneos>	Regra = “22” Máximo = “2”		Os docentes podem ser participantes de no máximo dois programas de pós-graduação.
<Aluno>			Tag referente aos Alunos da pós-graduação da UFG.
<Matrícula Simultânea>	Regra = “27” Máximo = “1”		Todo aluno deve estar matriculado em um único programa em um dado instante de tempo.
<Disponibilidade>			Tag agrupadora da disponibilidade do software SISPG.
<SemanaInício>	Quando = "segunda-feira" Regra=“S2”		O software SISPG está disponível de segunda-feira a sexta-feira.
<SemanaFim>	Quando = "sexta-feira” Regra=“S2”		
<HoraInício>	TipoCampo = "Time" Regra=“S2”	07:30:00	O software SISPG está disponível das 07:30 até as 17:00.
<HoraFim>	TipoCampo = "Time" Regra=“S2”	17:30:00	
<MaxUsuários>	Regra = “S3” TipoCampo= "Integer"	100	O número máximo de acessos simultâneos no SISPG é 100.
<Participante>			Tag agrupadora da titulação dos participantes da pós-graduação da UFG.
<Cargo>	Regra = “19” TipoCampo = “String” Titulo = “doutor/ mestre/ graduação/ 1º grau/ nenhum”	Docente, Aluno, Pesquisa-dor, Outro participante	Regra que faz o mapeamento entre a função do funcionário ou aluno da UFG e o grau exigido da sua titulação.
<SolicitarRequerimento>			Tag referente à solicitação de requerimentos.

<TipoRequerimento>	Regra = “31” Autorização="S/N"	Contém os nomes documentos.	Esta tag contém os nomes do documentos disponíveis da UFG, e o seu atributo *Autorização* diz se o mesmo exige autorização do orientador para ser deferido.
<AcessoSISPG>			Tag referente aos acessos disponíveis no SISPG.
<SessãoExpira>	Regra=“S4” TipoCampo=“Integer” Unidade = “min”	10	A inoperabilidade do software SISPG por mais de 10 minutos, acarreta a expiração da sessão.
<Usuário>			Tag que contém as restrições de acesso ao SISPG por usuário.
<MaxAcesso>	Regra = “S5” TipoCampo= “Integer”	1	Cada usuário só pode se conectar ao SISPG uma única vez em um mesmo intervalo de tempo.

4. Aplicação de Manutenção das Regras

A análise das regras de negócio e a proposta de representação delas em XML é apenas parte deste trabalho. Neste capítulo é definida uma aplicação para que a manipulação das informações representadas em XML possa ser realizada por um usuário que não tenha conhecimento de XML.

4.1. Visão Geral

"Uma compreensão completa dos requisitos de software é fundamental para um desenvolvimento de software bem-sucedido." [ENGENHARIA]

A análise das Regras que regem o SISPG nos levou a considerá-las complexas, a tal ponto de projetarmos um software que gerenciasse a sua manutenção. O software foi batizado com o nome de Regras e suas funcionalidades e características estão apresentadas na seção subseqüente.

O software Regras permite que os valores das regras da pós-graduação da UFG sejam alterados, para que eles sempre representem o cenário atual da UFG. Por exemplo: suponha que o prazo máximo de entrega de algum requerimento solicitado seja revisado de 15 dias para 10 dias. Só será necessário alterar o valor desta regra através do uso do software Regras, cuja definição e implementação é proposta deste trabalho, o que irá reduzir consideravelmente as manutenções que envolvam os elementos voláteis das regras consideradas.

4.2. Requisitos Funcionais

Essa seção apresenta as características que deverão estar presentes no programa a ser construído, com uma descrição sucinta de seu objetivo e realização. A seção 4.4 fornece os requisitos funcionais, descritos em casos de uso, correspondentes a tais características.

4.2.1. Ler e salvar arquivos XML no disco

O software Regras deverá ser capaz de ler arquivos armazenados no disco que tenham a extensão XML. Somente serão lidos os arquivos XML que atenderem às especificações do seu DTD correspondente, arquivos com formatos inválidos não poderão ser lidos e interpretados.

O software Regras também deverá ter a funcionalidade de salvar as alterações efetuadas pelo usuário em um arquivo com a extensão XML.

4.2.2. Excluir elementos de Regras do arquivo XML

O software Regras deverá gerenciar o processo de exclusão de elementos do arquivo XML, com base nas definições do DTD, ou seja, só poderão ser excluídas as tags que não estão configuradas como obrigatórias.

4.2.3. Inserir elementos de Regras no arquivo XML

O software Regras deverá gerenciar o processo de inserção de elementos de Regras no arquivo XML, com base nas definições do DTD correspondente.

Só poderão ser inseridos elementos já especificados no DTD.

Um elemento só poderá conter mais de um filho com o mesmo nome, se no seu DTD correspondente, estiver especificado que ele pode ocupar mais de um valor.

4.2.4. Alterar elementos de Regras no arquivo XML

O software Regras deverá permitir que os valores e atributos dos elementos do arquivo XML possam ser alterados, com base nas definições do DTD correspondente.

Cada elemento de regra tem um atributo que identifica o tipo de campo que ele pode armazenar. Portanto, esta função que permite a alteração de valores das regras deve validar todas as solicitações de alteração, de modo que o arquivo XML esteja sempre bem formado.

4.2.5. Garantir conformidade do arquivo XML com o seu DTD

O software Regras deverá garantir que todo arquivo XML que tenha sido manutenido por ele, esteja sempre bem formado e em conformidade com o DTD especificado.

4.2.6. Gerar um arquivo XML padrão

O software Regras deverá gerar um arquivo XML "padrão", com o propósito de lidar com situações excepcionais.

4.3. Requisitos Não Funcionais

Essa seção apresenta os requisitos não funcionais eliciados com uma descrição sucinta de seu objetivo e realização.

4.3.1. Fácil manutenção

O software deve ser de fácil manutenção, para acomodar mudanças nas regras propriamente ditas e que, neste caso, exigem alteração correspondente neste programa.

4.3.2. Facilitar o aprendizado e o uso

O software Regras foi desenvolvido, tendo-se em mente, a preocupação em manter uma interface amigável, composta por menu, janelas e teclas de atalho, o que deverá facilitar a sua utilização.

Porém, isto não dispensa o usuário de conhecer as regras que regem a pós-graduação da UFG.

4.3.3. Confiabilidade das informações

O software Regras deve garantir que o valor da regra armazenado no arquivo XML esteja consistente com as informações fornecidas.

4.4. Diagramas de Casos de Uso

Essa seção apresenta os casos de uso do sistema Regras, organizados em diagramas que demonstram a forma como estes se inter-relacionam e, posteriormente, uma descrição da funcionalidade de cada caso de uso.

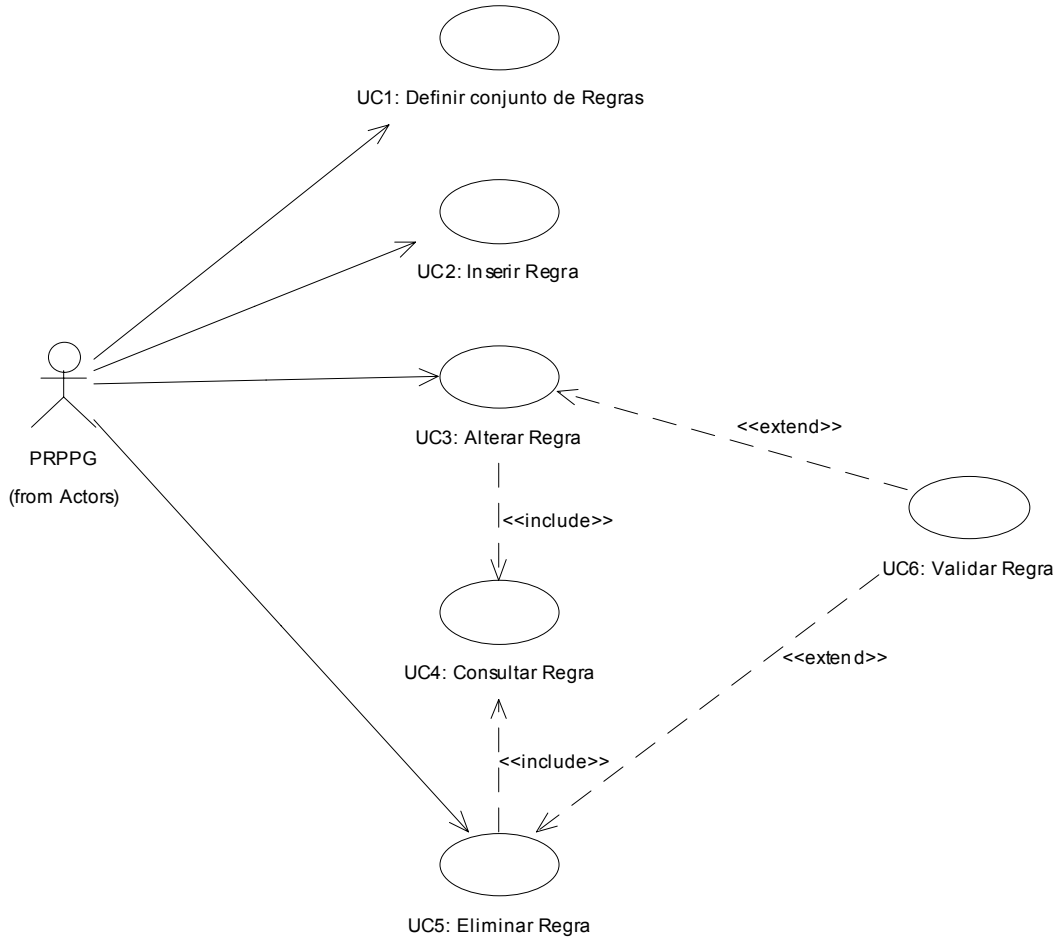


Figura 4.1: Diagrama de casos de uso do software Regras

4.4.1. Descrição dos Casos de Uso

Essa seção descreve, de forma individual, o fluxo dos casos de uso do software Regras.

a) UC1: Definir conjunto de Regras

Através deste caso de uso, o ator PRPPG poderá criar conjuntos de regras. Cada conjunto de regra é armazenado isolado um dos demais.

a.1.) Fluxo de Eventos:

1. O ator PRPPG deseja definir um novo conjunto de regras;
2. O sistema requisita um nome para o novo conjunto;
3. O ator PRPPG fornece o nome e requisita criação;
4. O sistema, conforme as restrições a serem observadas na criação de um conjunto de regras (DTD), cria um conjunto padrão de regras;
5. O usuário é informado da criação e o caso de uso termina.

b) UC2: Inserir Regra

O ator PRPPG poderá inserir novas regras ao conjunto de regras já configurado. Qualquer alteração nas regras é validada, de modo a garantir que o arquivo que contém as regras estará sempre em conformidade com o DTD especificado.

b.1.) Fluxo de Eventos:

1. O ator PRPPG escolhe regra a ser inserida;
2. Dependendo da restrição da regra, o sistema requisita um valor e/ou atributo para a nova regra;
3. PRPPG fornece valor e/ou atributo para a nova regra;
4. O sistema valida a inserção da nova regra, de acordo com as restrições do DTD;
5. O caso de uso termina.

b.2.) Fluxo Alternativo:

Inserção efetuada

Regra inserida está em conformidade com o DTD, e a alteração efetuada é automaticamente salva no disco.

Inserção não efetuada

Tentativa de inserção de regra não está em conformidade com o DTD. O arquivo XML permanece inalterado e o usuário é informado.

c) UC3: Alterar Regra

Através deste caso de uso, o ator PRPPG poderá efetuar as manutenções desejadas no valores e atributos configurados para as regras.

c.1.) Fluxo de Eventos:

1. O ator PRPPG escolhe regra a ser alterada;
2. PRPPG informa novo valor para a regra;
3. O sistema valida se o valor a ser atribuído a regra é válido, no que diz respeito ao tipo do campo, exemplo: campo do tipo string, inteiro, time, etc;
4. PRPPG informa novo atributo para a regra, de acordo com a lista de atributos disponíveis para a regra selecionada.
5. O caso de uso termina.

c.2.) Fluxo Alternativo:

Alteração efetuada

Alteração de valor da regra está em conformidade com o DTD, e o "novo" arquivo XML é automaticamente salvo no disco.

Alteração não efetuada

Tentativa de alteração de valor configurado para a regra não está em conformidade com o DTD. O arquivo XML permanece inalterado e o usuário é informado.

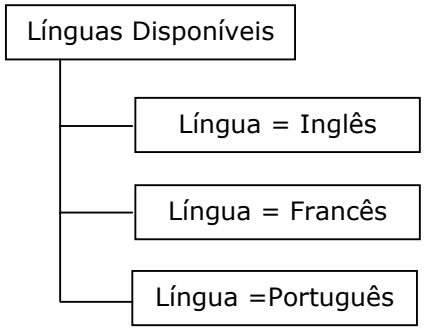
d) UC4: Consultar Regra

O ator PRPPG poderá consultar as regras que regem a pós-graduação da UFG, visualizar seus valores e atributos.

As regras estão agrupadas de acordo com a sua semântica, o que facilitará a localização de determinada regra dentro do conjunto de regras configuradas.

e) UC5: Eliminar Regra

O ator PRPPG poderá eliminar os valores das regras que considerar que não fazem mais parte do cenário da UFG, lembrando que, só poderão ser excluídas as regras que se repetem no conjunto de regras. As regras que são únicas não poderão ser excluídas. Observe a figura abaixo, para visualizar casos de exclusões válidas e inválidas.



Observe que, na tag Línguas Disponíveis, existem 03 (três) regras Língua, com valores distintos.

Duas das tags Língua poderão ser excluídas. Já a tag Línguas Disponíveis não poderá ser excluída, pois ela é única no arquivo XML.

Figura 4.2: Representação de exclusão de Regras

e.1.) Fluxo de Eventos:

1. Ator PRPPG escolhe regra a ser excluída;
2. Sistema valida se regra pode ser excluída;
3. O caso de uso termina.

e.2.) Fluxo Alternativo:

Exclusão efetuada

Exclusão de regra foi validada, ou seja, arquivo continua em conformidade com o DTD. O arquivo XML é automaticamente salvo no disco.

Exclusão não efetuada

Tentativa de exclusão de regra não está em conformidade com o DTD. O arquivo XML permanece inalterado e o usuário é informado.

f) UC6: Validar Regra

Qualquer alteração a ser efetuada no arquivo XML, primeiramente será validada pelo software Regras, com o objetivo de garantir a sua conformidade com o arquivo DTD correspondente.

Por exemplo, caso o usuário deseje alterar o valor ou o atributo de alguma regra, deverá selecioná-la e informar o novo valor ou atributo desejado. O sistema valida se a alteração é válida, no que diz respeito à tipo de campo e tamanho de campo, e salva a alteração no arquivo XML.

Caso o usuário deseje excluir alguma regra do arquivo XML, o sistema validará se esta tag poderá ser excluída, caso contrário, o sistema emitirá uma mensagem ao usuário, impossibilitando a exclusão.

4.5. Modelo de Domínio

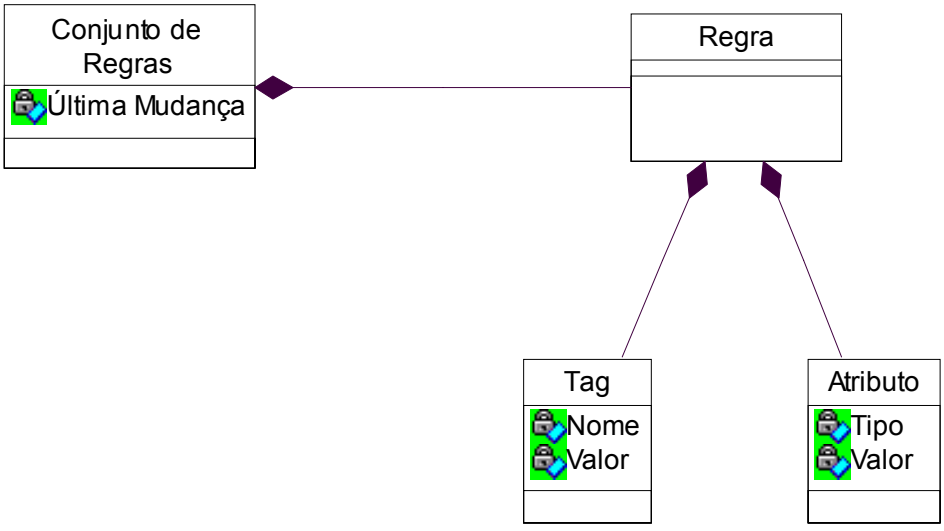


Figura 4.3: Modelo de Domínio do software Regras

5. Protótipo da Aplicação de Manutenção das Regras

Este capítulo pretende apresentar todas as funcionalidades internas do software desenvolvido, com o objetivo de facilitar o processo de manutenção e evitar os erros oriundos desta atividade.

5.1. Apresentação das classes que compõem o projeto

O software de manutenção das regras é composto por 09 (nove) classes, apresentadas abaixo:

AdapterNode	É a classe responsável em analisar o nó DOM, e retornar o valor do seu texto correspondente.
AlterarRegra	Aqui está definida a interface para se efetuar alterações no valor ou no atributo da regra armazenada no arquivo XML.
DomToTreeModelAdapter	Classe responsável em converter o documento (document) em uma árvore (componente JTree).
InserirRegra	Esta classe contém a interface gráfica para inserção de novos nós na árvore.
MenuPrincipal	Como o próprio nome sugere é a classe principal do programa. Aqui estão definidos os métodos para ler o arquivo XML, validar as alterações no arquivo XML de acordo com o arquivo DTD correspondente, todos os métodos referentes às manutenções no arquivo, dentre várias outras funcionalidades.
ParserErrorHandler	Nesta classe está a implementação da validação do parse no documento, para evitar que um arquivo com formato inválido possa ser manutenido pelo software Regras.
ValidateXMLAction	Esta classe foi de extrema importância no desenvolvimento do software Regras, pois permitiu a validação das alterações solicitadas pelo usuário.
XmlFileFilter	Esta classe especifica que a única extensão de arquivo considerada pelo software de manutenção das regras é XML, assim, só poderão ser abertos ou salvos arquivos do tipo XML.
XMLUtils	Possui dentre outras funcionalidades, o objetivo de salvar o documento que está sendo mostrado na tela em um arquivo XML.

Figura 5.1: Representação das classes do software Regras

5.2. Detalhamento das classes empregadas

5.2.1. Classe AdapterNode

Objetivos principais

1. Analisar o nó DOM e retornar o texto correspondente que será mostrado na árvore.
2. Esta classe também retorna os filhos, os índices e a quantidade de nós filhos.

Métodos definidos

AdapterNode(Node node)

Método responsável em construir um nó `Adapter` através de um nó DOM.

toString()

Método que retorna uma string que identifica o nó na árvore.

content()

Método que retorna uma string que representa o valor do nó selecionado na árvore, para os nós do tipo: `TEXT_TYPE`.

index(AdapterNode child)

Método que recebe um `AdapterNode` como parâmetro e retorna um valor inteiro, que corresponde à quantidade de filhos do nó especificado.

child(int searchIndex)

Este método recebe como parâmetro um valor inteiro, que representa o número do nó correspondente; localiza este nó no documento e instancia o método `AdapterNode(node)`, que irá converter o nó em um tipo `AdapterNode`, que será adicionado na árvore. Este método não considera os nós que são do tipo `ELEMENT_TYPE`.

convertNode()

Este método converte o nó do tipo AdapterNode em um do tipo Node, que é o tipo que deve ser utilizado para se efetuar manutenções no documento.

childCount()

Método responsável em retornar a quantidade de nós do tipo ELEMENT_TYPE.

5.2.2. Classe AlterarRegra

Objetivos principais

1. Permitir que o usuário possa alterar o valor de uma regra ou o atributo nela contida.

Métodos definidos

AlterarRegra (String pNomeRegra, String pValorRegra, String[][] pAtribRegra, JFrame frame)

Método constructor recebe como parâmetro o nome da regra, o valor atualmente armazenado na regra, uma matriz que contém o mapeamento do atributo atualmente armazenado na regra e o componente Frame da Classe MenuPrincipal.

Caso o usuário clique no botão de Cancelar nenhuma alteração é realizada, porém, quando o usuário alterar o valor ou atributo armazenado na regra, será chamado o método alterarNo(auxValorNo, auxAtributoNo) responsável em efetuar as alterações.

5.2.3. Classe DomToTreeModelAdapter

Objetivos principais

1. Converter o documento (que é do tipo DOM) em um modelo hierárquico de árvores.

Métodos definidos

Os métodos definidos nesta classe não precisam ser redefinidos, mesmo em caso de alteração do arquivo XML.

5.2.4. Classe InserirRegra

Objetivos principais

1. Permitir que uma nova regra seja inserida no arquivo XML.

Métodos definidos

InserirRegra (String [] pfilhosArray, JFrame frame)

Método constructor recebe como parâmetro um vetor que contém os filhos do nó selecionado pelo usuário, ao qual se deseja inserir novos elementos e o componente Frame da classe MenuPrincipal.

Caso o usuário selecione um nó e clique em OK, é instanciado o método addNode(String) da classe MenuPrincipal, que efetua a inserção do nó desejado.

Caso o usuário clique em Cancelar, nenhuma alteração é feita no arquivo XML.

5.2.5. Classe MenuPrincipal

Objetivos principais

1. Ler o arquivo XML, e analisá-lo de acordo com a sintaxe definida no DTD.
2. Criar uma estrutura correspondente na memória, e apresentar os dados em forma de hierarquia.
3. Permitir manutenção das Regras.

Métodos definidos

MenuPrincipal()

No método constructor foram criados os componentes visuais de interface com o usuário (por exemplo: o menu, os botões para alteração no arquivo XML, e a frame que contém estes componentes).

main(String args[])

Aqui está sendo criada uma instância da classe MenuPrincipal.

openFile()

Método responsável em abrir uma janela para que o usuário escolha o arquivo que deseja abrir, só será possível abrir arquivos com a extensão XML.

saveFile()

Laço que permite que o usuário salve no disco o arquivo que está sendo alterado, a extensão do arquivo necessariamente deverá ser XML.

mostrarXML ()

Método responsável em representar os nós em uma árvore de hierarquias (componente JTree), através da instanciação da classe DomToTreeModelAdapter.

O programa só será capaz de mostrar na tela os arquivos XML válidos, ou seja, que estejam em conformidade com o DTD especificado. Caso o arquivo esteja com o formato inválido, será emitida uma mensagem de erro ao usuário.

Cada nó que é selecionado na árvore é armazenado em uma variável global, de modo que os métodos de inserção, alteração e remoção possam sempre obter qual o nó que está com o foco na árvore.

alterarData()

Método responsável em atualizar a data e a hora que o usuário alterou pela última vez as regras da pós-graduação da UFG. Esta atualização é realizada na tag ÚltimaAlteração, que necessariamente tem que estar localizada na primeira posição abaixo do nó Root (ou seja o nó pai – Regras).

A data e a hora a serem atualizadas na tag ÚltimaAlteração são obtidas do Sistema Operacional da máquina onde a alteração foi realizada.

alterarNo(String novoValorNo, String novoAtributoNo)

Método responsável em controlar as alterações nos valores e atributos das regras. Caso o parâmetro novoValorNo recebido pelo método seja diferente de vazio, o método setValorNo(novoValorNo) será executado; caso o parâmetro novoAtributoNo recebido pelo método seja diferente de vazio, o método setAtributeNo(novoAtributoNo) será executado.

setValorNo(String valor)

Método recebe um valor String, que deverá ser atribuído ao valor do nó selecionado na árvore. O valor do nó selecionado está disponível em uma variável do tipo Node, que é atualizada no método mostrarXML().

Antes de fazer a alteração solicitada pelo usuário, o sistema valida se o tipo de campo que o usuário digitou é válido (por exemplo, não permite que seja digitado um valor alfa-numérico em um campo que só aceita valores inteiros). Para fazer esta validação, é feita uma consulta no método tipoCampo, que retorna a restrição para o tipo do campo desejado.

Este método é capaz de validar campos dos seguintes tipos: string, inteiro, decimal e time. Caso haja necessidade de se validar algum tipo de campo diferente dos acima citados, basta criar um novo método de validação para o tipo de campo desejado (conforme os quatro métodos abaixo apresentados).

Após a alteração do valor do nó, o método alterarData() é executado, com o objetivo de atualizar a data de última alteração do arquivo XML.

validateString(String field)

Método recebe como parâmetro um campo String, e valida se os caracteres nele contidos correspondem a um campo string válido.

Foram considerados como caracteres válidos os seguintes caracteres:
"abcdefghijklmnopqrstvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
123456789áÁéÉíÍóÓúÚâÂêÊôÔãÃçÇ".

validateInteger(String field)

Método recebe como parâmetro um campo String, e valida se os caracteres nele contidos correspondem a um campo inteiro válido.

Foram considerados como inteiros válidos os seguintes caracteres: "0123456789".

validateFloat(String field)

Método recebe como parâmetro um campo String, e valida se os caracteres nele contidos correspondem a um campo decimal válido.

Foram considerados como campos float válidos os seguintes caracteres: "0123456789,".

validateTime(String field)

Método recebe como parâmetro uma hora String, e valida se a mesma é válida.

setAtributeNo(String valor)

Método que permite a alteração no atributo do nó selecionado. O valor do nó selecionado está disponível em uma variável do tipo Node, que é atualizada no método mostrarXML().

Os nós do arquivo XML deste trabalho foram projetados para ter os atributos default's: TipoCampo e Regra, além dos atributos específicos de cada nó. Este método só permite que sejam alterados os atributos específicos dos nós, portanto não será possível alterar o número da regra e o tipo do campo pela manutenção aqui desenvolvida.

Os elementos abaixo especificados exemplificam os atributos padrões e os específicos de duas regras contempladas neste projeto.

Exemplo 1: o mapeamento abaixo corresponde à Regra número 8, que representa o tempo máximo para inscrever em disciplina, o valor nele contido é do tipo Inteiro, e a Unidade do Valor contido é em Dias.

```
<Tempo>
      <Disciplina>
         <Inscrever Regra = "8" Unidade = "dia" TipoCampo = "Integer">1</Inscrever>
      </Disciplina>
</Tempo>
```

Algumas regras especificadas no nosso software não possuem o atributo tipo do campo, observe o exemplo abaixo:

Exemplo 2: o elemento abaixo corresponde à Regra número 23, que especifica em qual situação, o estágio docência é obrigatório. Neste mapeamento não foi especificado o atributo Tipo do Campo, pois esta regra não tem um valor definido, ela somente possui atributos.

```
<Obrigatório>
      <EstágioDocencia>
         <Instituição Regra = "23" Unidade = "S">UFG</Instituição>
      </EstágioDocencia>
</Obrigatório>
```

Restrição: Foi definido que o software Regras irá alterar somente um atributo de cada nó, por mais que ele possa conter mais de um atributo. Veja abaixo:

O elemento Corrigir Dissertação possui 3 atributos: o número da regra, o tipo do campo e a unidade. Porém, somente o atributo Unidade poderá ser alterado pelo software Regras.

```
<CorrigirDissertação Regra = "34" Unidade = "dia" TipoCampo = "Integer"> 30
</CorrigirDissertação>
```

Após a alteração do valor do nó, o método alterarData() é executado, com o objetivo de atualizar a data de última alteração do arquivo XML.

removeCurrentNode()

Método que tenta efetuar a operação de exclusão do nó selecionado pelo usuário. O valor do nó selecionado está disponível em uma variável do tipo Node, que é atualizada no método mostrarXML().

Antes de remover o nó, o programa valida se o mesmo pode ser excluído, através da sua definição no DTD. Caso o nó seja obrigatório, o programa emite uma mensagem de que o nó não pode ser excluído.

Depois de atualizar o documento, o programa executa o método mostrarXML(), que atualiza todas as alterações na tela visível ao usuário.

addNode(String nomeNo)

Este método recebe como parâmetro uma variável string que contém o nome do nó que se deseja inserir no documento XML. É feita uma busca no documento até se encontrar a posição que se deseja inserir o nó, obedecendo o critério alfabético.

Antes de adicionar o novo nó, o programa valida se as definições para o elemento no arquivo DTD permitem que o nó possua mais de um elemento com o mesmo nome. Caso o nó seja único, o programa emite uma mensagem de que o nó selecionado não pode ser inserido no documento.

Caso a atualização no documento possa ser efetuada, o programa executa o método mostrarXML(), que atualiza a tela visível ao usuário.

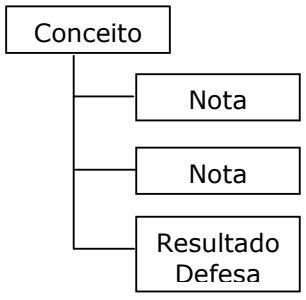
voltarDocument()

A função deste método é voltar o documento e o layout da árvore à posição anterior, caso as alterações solicitadas pelo usuário não possam ser efetuadas, devido às restrições especificadas no DTD.

As alterações solicitadas pelo usuário são executadas em um arquivo XML temporário. Após a alteração é feita a validação do arquivo, caso ele contenha algum erro, o programa executa o parse (leitura e interpretação) do arquivo XML original, visto que neste ponto ele ainda não foi alterado e este arquivo volta a ser o arquivo utilizado no software Regras.

getNodeFilhos (String elemPath)

Este método recebe como parâmetro o nome do pai do nó selecionado na árvore, e retorna um vetor com os nomes de seus nós filhos. Caso o nó filho se repita, só será trazido o primeiro. Observe a figura abaixo:



Suponha que o elemento passado como parâmetro para o método getNodeFilhos seja Conceito. O resultado do método será um vetor de duas posições que contenha o elemento Nota e o elemento ResultadoDefesa.
(Vetor Resultado = [Nota, Resultado Defesa])

Figura 5.2: Exemplificação de elementos com tags repetidas

save(File file, boolean indenting, String publicID, String systemID)

Método responsável em salvar o documento no disco, através da instanciação do método serialize da classe XMLUtils.

mostrarAtributo (Node no)

Este método recebe como parâmetro o nó selecionado na árvore, busca os seus atributos (tipo do nó = ATTRIBUTE_NODE), e os joga no painel atributoPanel, situado no canto inferior direito da tela do software Regras.

O atributo TipoCampo não é mostrado na tela, ele só é empregado nas rotinas internas do software, para validação das solicitações de alterações feitas pelo usuário.

mostrarValorRegra (Node no)

Este método recebe como parâmetro o nó selecionado na árvore, busca o seu valor e o joga no painel valorPanel, situado no canto superior direito da tela do software Regras.

especificarAtributo (String parentPath)

Este método foi empregado para mapear os atributos armazenados no arquivo XML para uns que pudessem ser mais facilmente entendidos pelos usuários.

Foram criadas matrizes que agrupavam os mesmos tipos de atributos. Cada matriz possui duas colunas, sendo que a primeira contém o valor do atributo que será mostrado na tela do software Regras e a segunda contém o valor do atributo armazenado no arquivo XML.

O mapeamento dos atributos trouxe uma série de vantagens ao software Regras:

- Evitar erros de digitação do usuário.
- Maior interatividade e facilidade para alterar o atributo das regras.

Abaixo são apresentados os tipos de atributos contemplados pelo software Regras:

- Matriz que contém as Unidades ⇒ utilizada para armazenar as unidades de tempo e percentual. Observe a matriz:

unidadeArray[][] = {{"Ano", "ano"}, {"Mês", "mês"}, {"Dia", "dia"}, {"Percentual", "%"}}.

- Matriz para as quantidades ⇒ utilizada para armazenar as quantidades máximas de alguma regra. Por exemplo, a quantidade máxima de vezes que um aluno pode trancar um curso de pós-graduação (regra 25).

qtdMaxArray[][] = {{"1", "1"}, {"2", "2"}, {"3", "3"}, {"4", "4"}}.

- Matriz para os valores booleanos (Sim e Não) ⇒ utilizada para fazer o mapeamento entre as regras que possuem valor booleano ("S" e "N").

valorArray[][] = {{"Sim", "S"}, {"Não", "N}}.

- Matriz que contém as unidades de tempo ⇒ utilizada para fazer o mapeamento das regras que possuem atributos de tempo.

tempoArray[][] = {{"Hora", "hora"}, {"Minuto", "min"}, {"Segundo", "seg"}}.

- Matriz que contém o mapeamento dos conceitos de notas ⇒ utilizada pela regra número 2.

conceitoArray[][] = {{"A", "A"}, {"B", "B"}, {"C", "C"}, {"D", "D"}, {"E", "E"}}.

- Matriz que contém os dias da semana ⇒ utilizada para armazenar dias da semana. Observe a matriz:

diaSemanaArray[][] = {{"segunda", "segunda-feira"}, {"terça", "terça-feira"}, {"quarta", "quarta-feira"}, {"quinta", "quinta-feira"}, {"sexta", "sexta-feira"}, {"sábado", "sábado"}, {"domingo", "domingo"}}.

- Matriz que contém o mapeamento do status das disciplinas ⇒ utilizada pela regra número 7.

statusArray[][] = {{"Obrigatória", "obrigatória"}, {"Optativa", "optativa"}}.

- Matriz que contém o mapeamento do título dos participantes da pós-graduação da UFG ⇒ utilizada pela regra número 19.

tituloArray[][] = {{"Doutor", "doutor"}, {"Mestre", "mestre"}, {"Graduação", "graduação"}, {"1º Grau", "1 Grau"}, {"Nenhum", "nenhum}}.

- Também foi criada uma matriz vazia, para que não seja mostrado nenhum atributo para as regras que não o possuem.

vazioArray[][] = {{"",""}}.

Caso seja necessário criar uma nova matriz que contenha vários elementos, talvez seja necessário aumentar a largura do layout da classe AlterarRegra, para que ela comporte a nova quantidade de elementos.

Caso seja necessário configurar uma nova regra que tenha um atributo semelhante aos já existentes, pode-se alterar a matriz para conter o novo valor. Por exemplo, temos o vetor de Unidades que contém os valores: ano, mês, dia e percentual, mas, podemos incluir, caso necessário, o novo valor semestre nesta matriz.

Criação de novas matrizes exigem que sejam feitas as seguintes alterações no código fonte do software Regras:

- Incluir a nova matriz no método constructor da classe MenuPrincipal;
- Incluir uma nova validação no método especificarAtributo, que contemple a nova matriz criada.

tipoCampo (Node no)

Método que recebe como parâmetro o nó selecionado na árvore, e procura se ele tem o atributo TipoCampo. O valor retornado por este método é justamente o tipo do campo definido para o nó especificado.

O resultado retornado por este método será útil no método setValorNo(String valor), para validar que o novo valor da regra está em conformidade ao tipo de valor que pode ser armazenado.

5.2.5.1. Opções de Menu disponíveis na Classe MenuPrincipal

Abrir Arquivo XML

Esta funcionalidade permite que seja aberto um arquivo da extensão XML que esteja salvo no disco.

O software só permite que sejam abertos arquivos válidos.

Caso o usuário solicite a abertura de um arquivo que não está em conformidade com o DTD, será emitida uma mensagem de Erro de Arquivo, e o mesmo não será aberto.

Ler Arquivo Padrão

Esta opção permite que um arquivo padrão seja salvo no disco. Este arquivo padrão está armazenado dentro de uma variável string armazenada internamente no projeto.

Esta funcionalidade é muito viável nos casos onde o usuário "perder" o arquivo XML utilizado.

Toda vez que o arquivo DTD for alterado será necessário atualizar a string que contém o arquivo XML, recompilar o projeto e distribuir a sua nova versão.

Se a string que contém o arquivo XML não está em conformidade com o seu DTD, será emitida uma mensagem de que o arquivo padrão está inválido, e o mesmo não será aberto.

5.2.6. Classe ParserErrorHandler

Objetivos principais

1. Validar se o parse (ou interpretação) do arquivo XML de acordo com o DTD deu certo.

Métodos definidos

hasErrors()

Este método retorna um valor booleano, de acordo com a validação do arquivo XML.

getMessage()

Método retorna mensagens detalhadas de erro, durante a tentativa de fazer o parse do arquivo XML.

Restrição: esta classe está preparada para validar qualquer tipo de arquivo XML que tenha o DTD como arquivo qualificador dos elementos. Porém, caso o XML esteja vinculado a um SCHEMA está classe não funciona.

5.2.7. Classe ValidateXMLAction

Objetivos principais

1. Validar se as solicitações de alteração no arquivo XML geram um arquivo em conformidade com o arquivo DTD especificado.

Passos executados nesta classe

1. As solicitações de alteração de alguma regra são automaticamente executadas no documento que está na memória.
2. O documento que está na memória é salvo no disco com o nome de `ValidarArq.xml`.
3. É feita uma instanciação da classe `ParserErrorHandler` que tenta fazer o `Parse` (ou interpretação) do arquivo, para verificar se ele está válido. Se o arquivo estiver válido, o método `Validar()` desta classe retorna o valor 1, caso contrário retorna o valor 0.

Restrição: esta classe está preparada para validar qualquer tipo de arquivo XML que tenha o DTD como arquivo qualificador dos elementos. Porém, caso o XML esteja vinculado a um SCHEMA está classe não funciona.

5.2.8. Classe XmlFileFilter

Objetivos principais

Classe que cria a extensão de arquivo XML, permitindo que somente sejam salvos ou abertos arquivos com a extensão XML.

Métodos definidos

Os métodos definidos nesta classe não precisam ser redefinidos, mesmo em caso de alteração do arquivo XML.

5.2.9. Classe XMLUtils

Objetivos principais

1. Salvar o arquivo XML no disco.

Métodos definidos

Os métodos definidos nesta classe não precisam ser redefinidos, mesmo em caso de alteração do arquivo XML.

Restrição: o método serialize, responsável em salvar o arquivo no disco funciona perfeitamente quando o arquivo XML é validado por um DTD, não foi testada a opção de salvar arquivo em disco que seja validado por um SCHEMA.

5.3. Manutenção no arquivo XML e no DTD

No desenvolvimento deste projeto, foi utilizado o editor XML SPY 4 para edição do arquivo XML. [XMLSPY4]

Caso seja necessário criar uma nova regra no arquivo XML já definido, ou mudar a restrição de algum elemento de regra, você também pode utilizar este editor.

5.3.1. Passos para alterar o arquivo XML e o DTD

Passos:

1. Clique com o botão contrário do mouse em cima do elemento que você quer alterar, e selecione *Adicionar filho/Elemento*.
2. Digite o nome do novo elemento.
3. Para incluir atributos no elemento criado, basta selecioná-lo e clicar em *Adicionar filho/Atributo*.
4. Para incluir nós filhos dentro do elemento criado, basta selecioná-lo e clicar em *Adicionar filho/Elemento*.
5. Após fazer todas as alterações necessárias, você terá que associar um novo arquivo DTD a este arquivo XML alterado. Vá ao menu *DTD/Schema* e escolha a opção *Gerar DTD/Schema*. Salve o arquivo DTD.
6. Algumas regras possuem um intervalo de dados fixos no DTD, e você terá que alterá-los. Atualmente, as regras que possuem intervalos fixos de atributos são:

   - O elemento *Cargo*: o arquivo DTD representativo do atributo *Titulo* do elemento *Cargo* deve conter os valores "doutor", "mestre", "graduação", "1Grau", "nenhum".

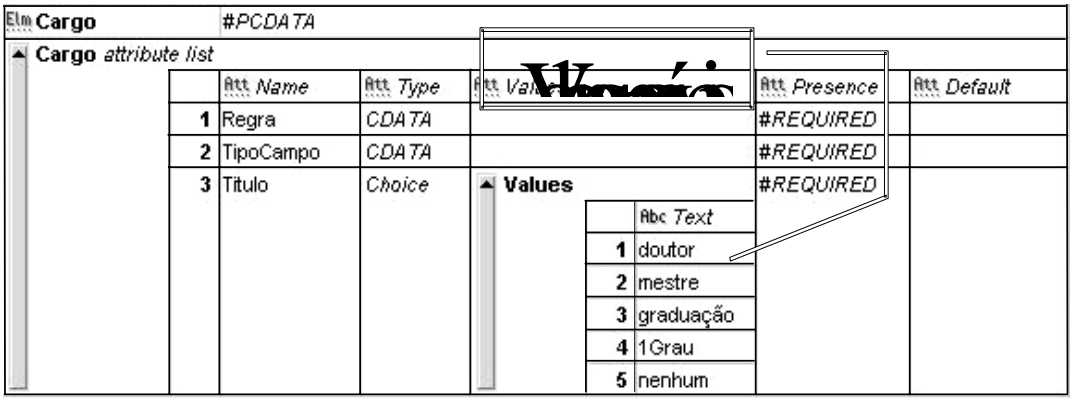


Figura 5.3: Alteração de Atributos no arquivo DTD (Regra 19)

Caso o arquivo DTD seja alterado, e você esqueça de fazer estas alterações, o software Regras emitirá erros, caso o usuário clique em alterar uma regra e selecione um atributo que não está definido no DTD.

Observe que os elementos definidos do DTD são os mesmos da figura abaixo, conforme já mostrado na explicação do método especificarAtributo da classe MenuPrincipal.

Figura 5.4: Alteração de Regra com Atributo Título

- O elemento Notas: o arquivo DTD representativo do atributo Valor do elemento Nota deve conter os valores "A", "B", "C", "D" e "E" (que são os mapeamentos de notas atualmente utilizados na UFG).

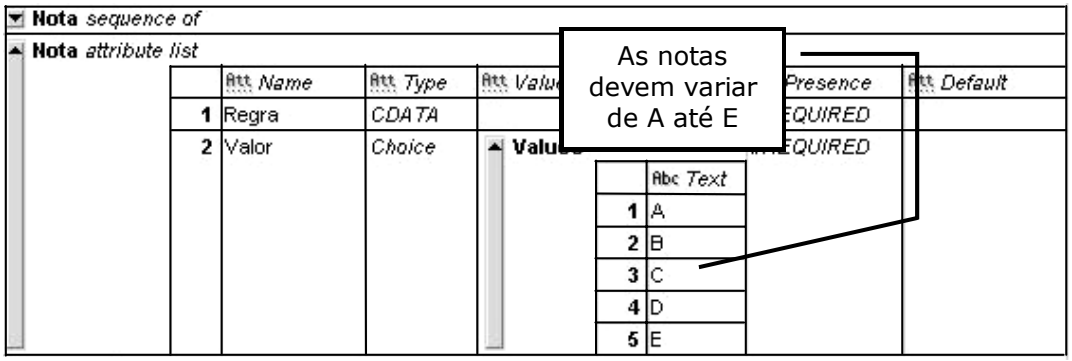


Figura 5.5: Alteração de Atributos no arquivo DTD (Regra 02)

- O elemento Instituição: o arquivo DTD representativo do atributo *Unidade* do elemento Instituição deve conter os valores "S" e "N".
- O elemento Nome da Disciplina: o arquivo DTD representativo do atributo *Status* do elemento Nome Disciplina deve conter os valores "obrigatória" e "optativa".
- O elemento Tipo de Requerimento: o arquivo DTD representativo do atributo *Unidade* do elemento Tipo Requerimento deve conter os valores "S" e "N".

6. Interface Gráfica do Software Regras

Neste capítulo serão apresentadas as telas que compõe a interface gráfica do software Regras.

A figura abaixo exemplifica as opções do Menu da Classe MenuPrincipal.

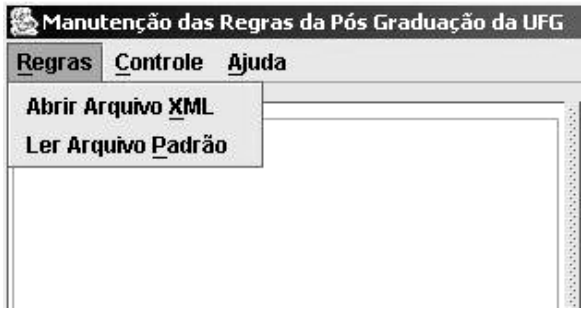


Figura 6.1: Opções do Menu da Classe MenuPrincipal

A figura abaixo exemplifica a solicitação de alteração da regra Inscrever.

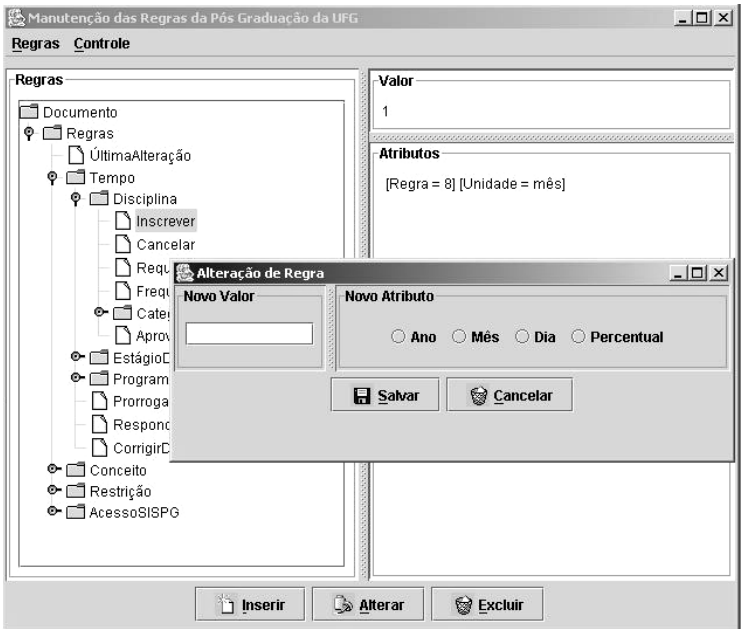


Figura 6.2: Exemplificação da Classe AlterarRegra

A figura abaixo representa a classe InserirRegra.

Quando o usuário selecionou o elemento Disciplina e clicou em Inserir, foi aberta uma nova janela, que contém os filhos do nó Disciplina.

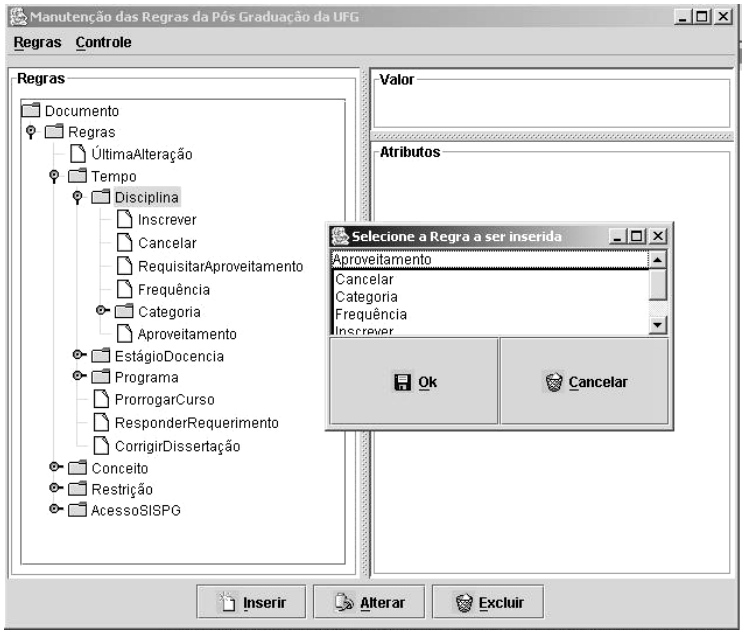


Figura 6.3: Exemplificação da Classe InserirRegra

O elemento Instituição possui o atributo "S". Quando clicamos em alterar o valor da regra, nos será aberta a janela abaixo, para que automaticamente escolhamos o novo atributo.

```
<Instituição Regra = "23" Unidade = "S">UFG</Instituição>
```

Figura 6.4: Alteração de regra com atributos booleanos (Sim ou Não)

O elemento SemanaInício possui o atributo “segunda-feira”. Quando clicamos em alterar o valor da regra, nos será aberta a janela abaixo, para que automaticamente escolhamos o novo atributo.

```
<Disponibilidade>
        <SemanaInício Quando="segunda-feira"/>
</Disponibilidade>
```

Figura 6.5: Alteração de regra com atributos dos dias da semana

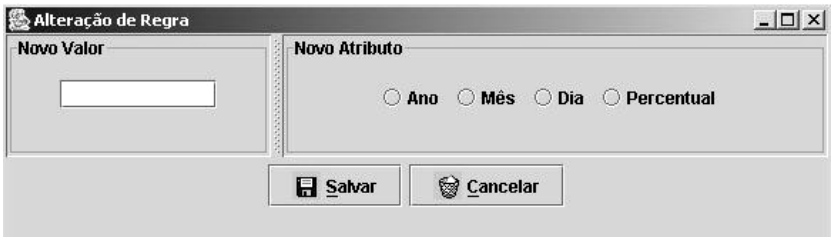


Figura 6.6: Alteração de regra com atributos das unidades

Figura 6.7: Alteração de regra que não possui atributos

7. Conclusão

Este trabalho está diretamente associado às futuras manutenções do SISPG, um sistema relativamente complexo que manipula significativo conjunto de regras que regem a pós-graduação da UFG.

O presente trabalho definiu um sistema e implementou um protótipo que auxilia o usuário a manipular as informações pertinentes às regras armazenadas em um arquivo XML. A definição do DTD correspondente e a própria organização do arquivo XML foi possível após uma análise das regras da pós-graduação da UFG.

O emprego da linguagem XML para mapeamento das regras da pós-graduação da UFG contribuiu para que cumpríssemos totalmente a proposta de desenvolvimento do software Regras, devido às suas vantagens, como: gera um arquivo fácil de ser interpretado, devido à organização das tags; permite fácil interpretação e validação do arquivo XML, através do DTD correspondente; possui o parse, além de ser portável.

A análise das regras permitiu identificar "elementos voláteis" que foram modelados como tags em um arquivo XML. Dessa forma, a aplicação produzida permite uma rápida alteração "consistente" deste arquivo que, de imediato, poderá ser consultado pelos componentes interessados do SISPG.

Trabalho futuro deverá validar o resultado obtido da perspectiva semântica das regras, e, eventualmente, proceder as respectivas mudanças no protótipo produzido.

Por último, entende-se que o presente trabalho é uma colaboração ao desenvolvimento do SISPG, além de ter proporcionado a oportunidade de solidificar, através das tecnologias empregadas, o conteúdo teórico fornecido no curso.

Índice Remissivo

A

análise 38
análise dos requisitos 13
arquivo padrão 58
árvore 47, 57
ator 42
atributo 53

C

características 38
casos de uso 41
cenário 23, 38
classes 46
componentes 13
conformidade 40

D

diagramas 41
dias da semana 56
documentação 24
DOM 47
DTD 15

E

Extensible Markup Language 17

F

funcionalidades 38

H

HTML 18

I

inserir 42
integridade dos dados 13
interface 50
interface gráfica 63
intervalos fixos de atributos 61

L

largura 57
layout 57

M

manutenção 40, 49
mapeamento 24
Mapeamento em XML 25, 26, 27, 29
matriz 57
método 55
Modelo de Domínio 45

N

nó 57
Node 52
notas 56
número da regra 53

P

parâmetro 52
parse 58
protótipo 66

Q

quantidades 56

R

Regras 38, 49, 55
regras de negócio 38
requisitos funcionais 38
requisitos não funcionais 40

S

salvar 39, 46
SCHEMA 59
semântica das regras 22
SISPG ..... xi, 14, 66
sistemas complexos 13
software 46, 53, 55

T

tag 21, 45
tempo 56
tipo do campo 53

U

UFG 57, 66
unidade 53

V

validar 57
ValidarArq.xml 59
voláteis 66

X

XML 15, 18, 38
XML SPY 4 60

Referências Bibliográficas

[CASOS DE USO] LUCENA, Fábio Nogueira. **Documento Casos de Uso do SISPG**. Disponível em: <http://software.inf.ufg.br/SISPG>. Acesso em: 15 ago. 2002.

[ENGENHARIA] PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995. 1056 p.

[JAVA XML] AHMED, Kal; ANCHA, Sudhir; CIOROIANU, Andrei; COUSINS, Jay; CROSBIE, Jeremy; DAVIES, John; GABHART, Kyle; GOULD, Steve; LADDAD, Ramnivas; LI, Sing; MACMILLAN, Brendan; RIVERS-MOORE, Daniel; SKUBAL, Judy; WATSON, Karli; WILLIAMS, Scott; HART, James. **Professional Java XML**. Canada: Wrox Press, 2001. 1158 p.

[REGRAS2002] LUCENA, Fábio Nogueira. **Regras que regem o SISPG**. Disponível em: <http://software.inf.ufg.br/SISPG>. Acesso em: 15 ago. 2002.

[REQUISITOS] CHUNG, L.; NIXON B. A.; YU E.; MYLOPOULOS, J. **Non-functional requirements in software engineering**. Boston: Kluwer Academic, 2000. 439 p.

[UML] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário**. 7 ed. Rio de Janeiro: Editora Campus, 2000. 472p.

[VISÃO2002] LUCENA, Fábio Nogueira. **Documento Visão**. Disponível em: <http://software.inf.ufg.br/SISPG>. Acesso em: 15 ago. 2002.

[WEB APPLICATIONS] MARUYAMA, Hiroshi; TAMURA, Kent; URAMOTO, Naohiko; MURATA, Makoto; CLARK, Andy; NAKAMURA, Yuichi; NEYAMA, Ryo; KOSAKA, Kazuya; HADA, Satoshi. **XML and Java™ – Developing Web Applications**. NJ: Addison-Wesley, 1999. 385 p.

[XML2003] BENEDITO, Miguel Furtado Júnior. **XML – Extensible Markup Language**. Disponível em: <http://www.gta.ufrj.br/grad/00_1/miguel/index.html>. Acesso em: 19 jan. 2003.

[XMLSPY4] ALTOVA. **XMLXPY 4**. Disponível em: <http://www.xmlspy.com>. Acesso em: 13 out. 2002.