# Xexam - uma linguagem de suporte para exames online (e-learning)

Daniel P. Soares
deps@mail.pt
DI/UM

M. Conceição Mota
conceicao_mota@aeiou.pt
DI/UM

José Carlos Ramalho
jcr@di.uminho.pt
DI/UM

10 de Fevereiro de 2003

**Resumo**

Hoje o E-learning, para além de ser um chavão utilizado por muitas instituições, também tem sido usado como estandarte de muitas instituições de ensino que se dizem na vanguarda.

Poderíamos começar a discutir o que é o E-Learning mas isso seria assunto para uma tese. Para o nosso trabalho, vamos considerar a definição mais consensual de ensino à distância onde se utiliza a Internet como veículo primordial para a comunicação entre professor e aluno e entre alunos. Nessa perspectiva, uma plataforma de E-Learning tem vários componentes: gestão de alunos e cursos, publicação de conteúdos, fóruns de discussão, chats, controle de acessos, diversas estatísticas e avaliação. É precisamente neste último componente que incide o trabalho descrito neste artigo. A avaliação online vai levantar sempre problemas de creditação mas discutir esse assunto não é o objectivo deste artigo.

Neste trabalho, apresenta-se uma linguagem XML — Xexam, que pode servir de suporte à criação de exames (por exemplo para auto-avaliação do aluno). Além da linguagem apresentam-se algumas ferramentas de suporte que fornecem algumas funcionalidades ao autor do exame: publicação em papel, publicação na Internet, correcção automática ao aluno que desenvolver o exame online e publicação do exame com correcção.

No fim, tecem-se algumas considerações sobre a integração desta aplicação numa metodologia de E-Learning.

## 1 Introdução

Actualmente, em Portugal, assiste-se a uma competição crescente entre as mais variadas instituições de ensino. Nesta competição, as instituições tentam destacar-se umas das outras pela qualidade. Esta qualidade tem-se vindo a traduzir, essencialmente, em dois aspectos: conteúdos e acessibilidade desses conteúdos.

Neste contexto, o modelo do E-Learning, já adoptado na Europa e América do Norte, está a sofrer uma grande adesão. Esta grande adesão está relacionada com alguns factores: a captação de alunos além das barreiras geográficas, a diminuição de custos ao nível da manutenção e suporte das instalações físicas e um maior controle, por parte do professor, das acções dos alunos.

No Departamento de Informática da Universidade do Minho, ao longo dos últimos oito anos, têm vindo a ser utilizadas algumas das metodologias de E-Learning (essencialmente, boas práticas tecnológicas): disponibilzação de conteúdos, afixação de notas e sumários, grupos de discussão e, iniciativas do tipo "o problema semanal"[RH95].

Nos últimos tempos, tem-se feito um grande esforço na automatização de algumas destas metodologias como, por exemplo, a produção de conteúdos.

Neste artigo, descreve-se um primeiro protótipo para uma possível automatização do módulo de avaliação.

O artigo está organizado num série de secções. Nas primeiras, descreve-se o conceito de avaliação no contexto apresentado e justificam-se as tecnologias que se irão utilizar no protótipo. Nas secções seguintes, descrevem-se as ferramentas desenvolvidas e que, no seu todo, constituem a implementação do protótipo. No fim, tiram-se algumas conclusões do trabalho desenvolvido, tecem-se algumas comparações com as iniciativas internacionais de normalização nesta área e, traçam-se algumas linhas para trabalho futuro.

## 2 Exames (online, e não só ...

À partida, por exame ”online”, quer-se dizer exames que estão disponíveis na Internet. Neste ponto, pode-se, desde já, fazer uma divisão: exames estáticos (o aluno apenas pode ler e/ou imprimir o exame) e exames dinâmicos (com interacção: o aluno pode resolver o exame, enviar a sua resolução e obter um feed-back desta). Mais à frente, na discussão da implementação do protótipo, veremos que estas duas classes de exames têm implicações tecnológicas distintas.

Quer o exame seja estático ou dinâmico e independentemente da tecnologia e da maneira de como irá ser disponibilizado, pode ainda ser mais categorizado como pertencente a um determinado tipo. Antes de avançar é importante detalhar cada um destes tipos. Assim, e recorrendo a uma classificação consensual no seio das entidades percursoras do E-Learning e numa perspectiva estrutural, temos os seguintes tipos de exame:

**Exame de escolha múltipla** – este tipo de exame é composto por uma sequência de questões em que cada questão é, por sua vez, composta por uma sequência de alíneas das quais apenas uma contem a resposta certa e é esta que o aluno deve assinalar.

**Exame de questões verdadeiro ou falso** – este tipo de exame é composto por uma sequência de questões em que cada questão é, por sua vez, composta por uma sequência de alíneas; cada alínea pode ser verdadeira ou falsa e é este valor lógico que o aluno deve indicar.

**Exame de questões de desenvolvimento** – este tipo de exame é composto por uma sequência de questões em que cada questão tem um enunciado que apresenta um problema; o aluno deverá desenvolver cálculos ou discursos que lhe permitam dar uma solução ao problema.

Numa perspectiva funcional, podemos ter os seguintes tipos de exame:

**Exame de uma tentativa** – neste tipo de exame o aluno dispõe apenas de uma tentativa (é o tradicional).

**Exame de várias tentativas** – neste tipo de exame o aluno dispõe de várias tentativas; depois de submeter o exame e perante o feed-back o aluno tem a oportunidade de alterar algumas respostas e reenviar o exame.

**Exame com tempo limite** – neste tipo de exame o aluno tem que resolver o exame dentro de um determinado limite temporal; o sistema de suporte monitoriza o tempo e após o limite dado interrompe a resolução do exame.

**Exame progressivo** – este tipo de exame corresponde a uma exame faseado; ao aluno é apresentada uma parte do exame e perante o resultado obtido o aluno poderá continuar a resolução do mesmo ou não.

**Exame com aleatoridade na ordem das questões** – neste tipo de exame, o sistema de suporte, sempre que um determinado exame é solicitado, baralha a ordem das questões antes de apresentar o exame ao aluno.

É claro que um exame não precisa de pertencer estritamente a um destes tipos. Podemos ter exames que são dum tipo híbrido apresentando características de vários tipos. Por exemplo: um exame que combina questões de desenvolvimento com questões de resposta múltipla ou de verdadeiro e falso.

Neste trabalho, e tratando-se de um protótipo que visa testar a tecnologia envolvida (que será descrita nas secções seguintes) no contexto que temos vindo a descrever, havia que limitar os tipos possíveis de exame. Por isso, os tipos de exame que foram considerados na implementação foram os de múltipla escolha, os de desenvolvimento e os correspondentes às combinações destes dois.

Na secção seguinte, apresenta-se a metodologia desenvolvida para a produção de exames.

## 3 Xexam

Normalmente, as plataformas de E-Learning são completamente abertas quanto aos formatos em que aceitam o conteúdos. Consequentemente, estes podem ser textos em HTML, animações em JAVA, documentos em PDF ou apresentações em PowerPoint.

No nosso grupo de trabalho, temos vindo a produzir conteúdos em XML [RH02] sempre que tal se justifique. Para isso, avaliamos sempre muito bem qual o resultado final pretendido.

Para a selecção do formato em que iriam ser produzidos os exames houve que ter em conta o resultado final pretendido. À partida, pretendiam-se quatro tipo de resultados:

1. Uma versão do exame em papel, igual à que tradicionalmente se utiliza nos exames presenciais. Este resultado é essencial para garantir a compatibilidade com o passado, evitando a criação de rupturas com o que os docentes vêm fazendo desde sempre. Se um docente, que está a usar este módulo, quiser produzir um exame tradicional para entregar em papel aos alunos o sistema deve permiti-lo sem qualquer custo adicional para o docente.

2. Uma versão online sem interacção. Equivalente à anterior só que em HTML. Para os casos em que o docente quer usar o sistema para publicar electronicamente o exame, após o exame ter sido realizado presencialmente ou para exames de auto-avaliação do aluno.

3. Um versão online com interacção. Neste caso, o exame é apresentado ao aluno sob a forma de um formulário HTML que aquele deverá preencher e enviar. Após o envio o sistema dá algum feed-back ao aluno: cotação obtida nas perguntas que foi possível corrigir automaticamente, marcação das respostas erradas com a respectiva solução, ...

4. Uma versão em papel, que corresponde à versão que fica armazenada para o professor, de um exame realizado online. Nesta versão, o exame é impresso com as perguntas e as respostas dadas pelo aluno.

Como vemos, logo à partida, queremos que o sistema produza quatro resultados distintos para o mesmo conteúdo. Sempre que tal acontece, estão criadas as condições para a utiliação de uma linguagem neutra, de anotação descritiva, para a produção de conteúdos.

Neste momento, a meta-linguagem XML é a solução adoptada quando este panorama se verifica.

Em termos mais formais, pode-se definir o XML como uma linguagem de anotação descritiva extensível, tendo, portanto, todas as características que tornam desejáveis este tipo de linguagens: independência relativamente às plataformas de software e de hardware utilizadas, longevidade, baixos custos de manutenção, facilidade na reutilização, ...

Para mais detalhes sobre o XML aconselhamos a leitura da vasta bibliografia existente, por exemplo [Wil01, HM01].

Como dissemos, o XML é uma meta-linguagem, não fornece, portanto, nenhum conjunto de marcas pré-definido. Fornece sim, os meios necessários à criação desses conjuntos de marcas os quais são designados por linguagens ou dialectos XML.

No nosso caso, vamos definir uma linguagem que nos permita realizar a anotação descritiva de um exame.

## 3.1 Definição da linguagem de anotação

Para definir uma liguagem de anotação há que criar o DTD (”Document Type Definition”) ou o Schema para essa linguagem. Esta fase corresponde à especificação da gramática se quisermos traçar um paralelo com as linguagens de programação.

Optou-se por definir um Schema devido ao suporte de Namespaces e às possibilidades de desenvolvimento modular que estes têm face aos DTDs.

Como já referimos, os tipos de exame que foram considerados neste projecto foram os exames de múltipla escolha, os de desenvolvimento e os correspondentes às combinações destes dois.

Assim, o Schema para exames é constituído por três partes:

**Cabeçalho** – onde se encontra toda informação relativa à metainformação que deverá estar associada ao exame, isto é, nome da disciplina, cursos a que se destina, ano lectivo, professor, chamada, data, hora, duração, sala e observações.

**Corpo** – onde o docente irá colocar as questões, que são o cerne de um exame; existem duas alternativas para a definição do corpo que estão relacionadas com tipo de exame que se pretende criar: um exame com apenas questões de múltipla escolha ou, um exame com questões de desenvolvimento e questões de mútipla escolha.

**Anexos** – onde o docente poderá colocar formulários, código, gráficos, etc...

Na apresentação dos vários elementos, utilizam-se figuras uma vez que o Schema é demasiado extenso para ser incluído neste artigo.

A figura 1 apresenta esquematicamente o elemento raiz para os documento do tipo exame.

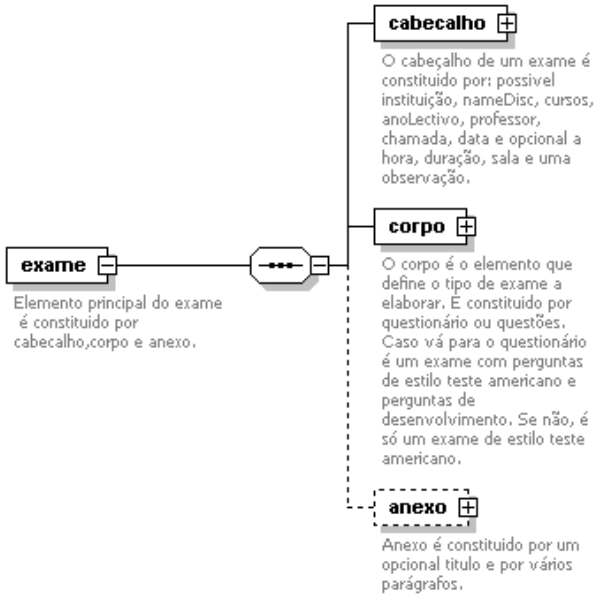


Figura 1: **Xexam: elemento raiz**

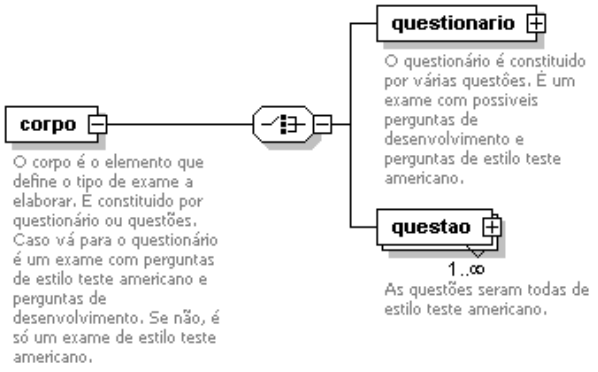


Figura 2: **Xexam: elemento corpo**

O corpo do exame é alternativamente composto por questões de múltipla escolha ou por um questionário em que estas poderão estar misturadas com questões de desenvolvimento (fig. 2).

A questão, por sua vez, é constituída por um enunciado e opcionalmente por um conjunto de alíneas (fig. 3).

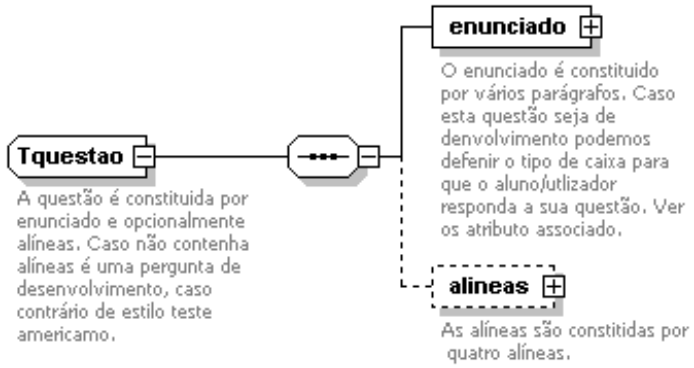


Figura 3: **Xexam: elemento questão**

Caso a questão seja de múltipla escolha tem que ter obrigatoriamente quatro alíneas em que cada alínea é constituída por um parágrafo (a resposta encerrada na alínea) e um atributo `validação` do tipo `boolean` onde o docente deverá assinalar a solução (fig. 4). As questões de desenvolvimento apenas têm um enunciado.

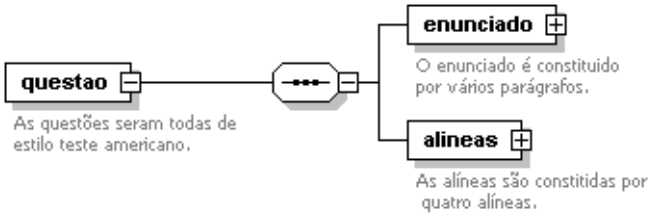


Figura 4: **Xexam: elemento questão**

Terminamos a discussão da linguagem XML apresentando o DTD (bem mais leve que o Schema), que ajudará a ilucidar o leitor no que respeita a elementos mais internos e aos atributos.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT exame (cabecalho, corpo, anexo?)>
<!ATTLIST exame
  nomeExame CDATA #REQUIRED>

<!ELEMENT cabecalho (instituicao?, nameDisc, cursos, anoLectivo,
                     professor, chamada, data, hora?, duracao?,
                     sala?, observacao?)>

<!ELEMENT corpo (questionario | questao+)>

<!ELEMENT anexo (titulo?, para+)>
<!ATTLIST anexo
  numeroA ID #REQUIRED>

<!ELEMENT cursos (curso+)>

<!ELEMENT observacao (para+)>

```

```
<!ELEMENT questionario (questao+)>

<!ELEMENT para (#PCDATA | destaque | ref | xref | lista_seq
                | figura | codigo | tabela)*>

<!ELEMENT curso (nome, ano, codigoDisc?)>

<!ELEMENT questao (enunciado, alineas?)>
<!ATTLIST questao
  numeroQ ID #REQUIRED>

<!ELEMENT alineas (alinea+)>

<!ELEMENT destaque (#PCDATA)>
<!ATTLIST destaque
  tipo (carregado | sublinhado | italico) #IMPLIED>

<!ELEMENT ref EMPTY>
<!ATTLIST ref
  destino IDREF #REQUIRED
  label CDATA #IMPLIED>

<!ELEMENT xref (legenda, url)>

<!ELEMENT lista_seq (item+)>
<!ATTLIST lista_seq
  tipo (numerada | nnumerada) #IMPLIED>

<!ELEMENT figura (legenda, grafico)>

<!ELEMENT tabela (linhas+)>
<!ATTLIST tabela
  titulo CDATA #REQUIRED
  numCol CDATA #REQUIRED>

<!ELEMENT enunciado (para+)>
<!ATTLIST enunciado
  tbox (pequena | media | grande) #IMPLIED>

<!ELEMENT alinea (para+)>
<!ATTLIST alinea
  validacao (true | false) #REQUIRED>

<!ELEMENT item (para+)>

<!ELEMENT grafico EMPTY>
<!ATTLIST grafico
  formato (jpeg | gif | bmp | wmf | png) #IMPLIED
  path CDATA #REQUIRED>

<!ELEMENT linhas (coluna+)>

<!ELEMENT coluna (#PCDATA | para)*>
```

Na secção seguinte, apresenta-se um documento XML exemplo criado de acordo com o Schema definido.

## 3.2 Exames em XML

Para ilustrar melhor a linguagem desenvolvida apresenta-se agora um exemplo de exame.

Este exame corresponde a um exame real dado aos alunos da licenciatura em relações internacionais.

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <exame
3      nomeExame="Xexam2.xml"
4      xsi:noNamespaceSchemaLocation="Xexam.xsd">
5    <cabecalho>
6      <instituicao>Universidade do Minho</instituicao>
7      <nameDisc>Políticas Económicas da União Europeia</nameDisc>
8      <cursos>
9        <curso>
10         <nome>Licenciatura em Relações Internacionais</nome>
11         <ano>4ºAno</ano>
12       </curso>
13     </cursos>
14     <anoLectivo>2001/2002</anoLectivo>
15     <professor>Dra. Sandrina Soares</professor>
16     <chamada>1ª Chamada</chamada>
17     <data>24 Janeiro</data>
18     <hora>9h30</hora>
19     <duracao>2 horas</duracao>
20     <sala>Sala 2209 e 2210</sala>
21     <observacao>
22       <para>Para cada questão selecione a afirmação verdadeira.</para>
23     </observacao>
24   </cabecalho>
25 ...
```

**Notas:**

**linha 1** – A linha 1 exibe a declaração XML indicando que conteúdo textual usa o alfabeto latino.

**linha 4** – A linha 4 associa o Schema desenvolvido a este documento. Um validador usa esta informação para garantir que o exame especificado está de acordo com a estrutura previamente estabelecida.

**linhas 5 a 24** – Nestas linhas, define-se a metainformação associada a este exame.

No bloco seguinte, apresenta-se a representação em XML de uma questão.

```
1  ...
2    <corpo>
3      <questionario>
4        <questao numeroQ="Q1">
5          <enunciado>
6            <para>No Tratado de Roma assinado em 1957 pelos seis
7            Estados-membros de então, ficava expresa a intenção em realizar:</para>
8          </enunciado>
9          <alineas>
10           <alinea validacao="false">
11             <para>uma união aduaneira;</para>
12           </alinea>
13           <alinea validacao="false">
14             <para>uma união aduaneira, um mercado comum e uma
15             união económica;</para>
16           </alinea>
17           <alinea validacao="true">
18             <para>uma união aduaneira e um mercado interno;</para>
19           </alinea>
20           <alinea validacao="false">
21             <para>um mercado comum, uma união económica e uma união
```

```
            económica e monetária.</para>
          </alinea>
        </alineas>
      </questao>
      ...
    </questionario>
  </corpo>
</exame>
```

Na secção seguinte, veremos como obter os vários resultados a partir deste tipo de documentos XML.

# 4 Geração das versões em papel e em HTML

Uma vez criado o exame em XML, torna-se necessário transformá-lo noutro formato conforme a utilização final. No início do artigo, discutiram-se os formatos pretendidos. Estes podem ser agrupados em dois grupos: papel e HTML. Para o papel optou-se por gerar LaTeX e deste PDF.

Assim, para obtermos a versão do exame em papel precisamos de uma transformação de XML para LaTeX, e para obtermos a versão Web, uma transformação de XML para HTML.

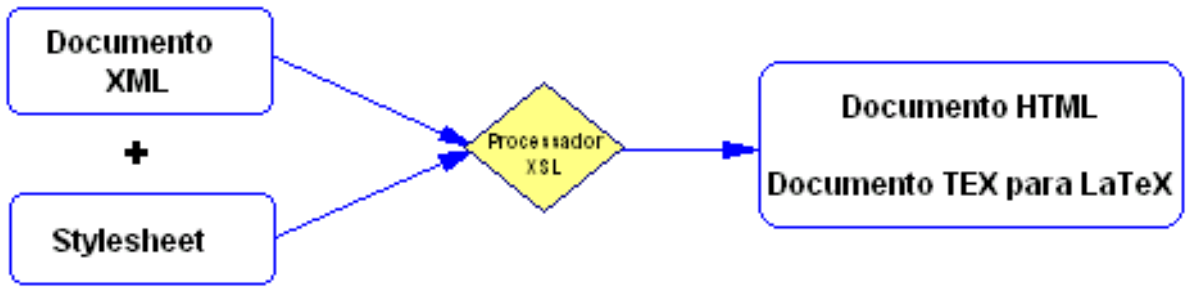


Figura 5: **Processo de Transformação**

De entre as várias ferramentas e correspondentes metodologias de transformação optou-se por utilizar XSL por várias razões: uma transformação é fácil de prototipar em XSL e, como uma folha de estilo XSL é um documento XML, pode ser transportada entre plataformas sem qualquer alteração. Assim, especificaram-se duas stylesheets que permitem a geração da versão HTML e da versão LaTeX a partir dos documentos XML já mencionados.

Dada a sua semelhança, apresenta-se a seguir, apenas a folha de estilo que transforma o exame numa página HTML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output encoding="ISO-8859-1" method="html"
              indent="yes" version="1.0"/>

  <xsl:template match="/">
    <html>
      <head>
        <title>
          <xsl:value-of select="cabecalho/nameDisc"/>
        </title>
      </head>
      <body style="MARGIN-LEFT: 10%; MARGIN-RIGHT: 10%;
                   FONT-FAMILY: 'Helvetica', 'Arial';">
```

```
18        <xsl:apply-templates/>
19      </body>
20    </html>
21  </xsl:template>
```

**Notas:**

**linhas 9-21** – Template responsável por criar a estrutura da página HTML.

```
22  <xsl:template match="cabecalho">
23    <h4 align="center">
24      <xsl:if test="instituicao">
25        <xsl:value-of select="instituicao"/>
26      </xsl:if>
27      <br/>
28      <xsl:value-of select="nameDisc"/>
29      <br/>
30    </h4>
31    <font size="2">
32      <center>
33        <xsl:apply-templates select="cursos"/>
34        <xsl:value-of select="anoLectivo"/>
35        <br/>
36        <xsl:value-of select="professor"/>
37        <br/><br/>
38        <xsl:value-of select="chamada"/>
39        <xsl:if test="data">
40          <xsl:text> (</xsl:text>
41          <xsl:value-of select="data"/>
42          <xsl:text>)</xsl:text>
43        </xsl:if>
44        <br/>
45        <xsl:value-of select="hora"/>
46        ...
47    </font>
48    <br/>
49    <hr color="#b0c4de"/>
50    <br/>
51  </xsl:template>
```

**Notas:**

**linhas 22-51** – Template que trata do cabeçalho; faz o a impressão estilizada da metainformação do exame; para os campos opcionais testa se estes existem antes de os processar.

```
52  <xsl:template match="corpo">
53    <xsl:choose>
54      <xsl:when test="//questionario">
55        <form action="http://localhost/cgi-bin/cgi2.cgi" method="post">
56          <fieldset style="BACKGROUND-COLOR: #f0f8ff">
57            <legend>Identificação do aluno:</legend>
58            <pre>
59              Nome:<input type="text" size="60" name="nome"/>
60              Curso:<input type="text" size="15" name="curso"/>
61              Numero:  <input type="text" size="15" name="numero"/>
62            </pre>
63          </fieldset>
64          <br/><br/><hr color="#b0c4de"/><br/>
65          <xsl:apply-templates/>
```

```
66          <p align="right">
67            <xsl:element name="input">
68              <xsl:attribute name="type">hidden</xsl:attribute>
69              <xsl:attribute name="name">exame</xsl:attribute>
70              <xsl:attribute name="value">
71                <xsl:value-of select="../@nomeExame"/>
72              </xsl:attribute>
73            </xsl:element>
74            <input type="reset" value="Limpar"/>
75            <xsl:text>  </xsl:text>
76            <input type="submit" value="Submit"/>
77          </p>
78        </form>
79      </xsl:when>
80      <xsl:otherwise>
81        <form action="http://localhost/cgi-bin/resultado.cgi" method="post">
82          <xsl:apply-templates/>
83          <p align="right">
84            <xsl:element name="input">
85              <xsl:attribute name="type">hidden</xsl:attribute>
86              <xsl:attribute name="name">exame</xsl:attribute>
87              <xsl:attribute name="value">
88                <xsl:value-of select="../@nomeExame"/>
89              </xsl:attribute>
90            </xsl:element>
91            <input type="reset" value="Limpar"/>
92            <xsl:text>  </xsl:text>
93            <input type="submit" value="Submit"/>
94          </p>
95        </form>
96      </xsl:otherwise>
97    </xsl:choose>
98  </xsl:template>
```

**Notas:**

**linhas 52-98** – Template que gera o formulário HTML.

**linhas 54-55** – Se o exame tiver perguntas de desenvolvimento, os dados fornecidos pelo aluno serão armazenados para posterior tratamento (na secção seguinte discutiremos a gestão da interacção através de CGIs – `cgi2.cgi`).

**linhas 81-82** – Se o exame for composto apenas por perguntas de múltipla escolha, então é possível dar um feedback imediato ao aluno sobre o seu desempenho – `resultado.cgi`.

```
99  <xsl:template match="questao">
100   <table border="1" width="100%" bgcolor="#f0f8ff">
101     <tr><td><b>
102       <xsl:element name="a">
103         <xsl:attribute name="name">
104           <xsl:value-of select="@numeroQ"/>
105         </xsl:attribute>
106         <i>Questão <xsl:number/>
107           <xsl:text>.   </xsl:text>
108         </i>
109       </xsl:element>
110       </b>
111       <xsl:text>   </xsl:text>
112       <xsl:apply-templates select="enunciado"/>
```

```
      </td>
      </tr>
    </table>
    <xsl:apply-templates select="alineas"/>
    <xsl:if test="not(alineas)">
      <p>
        <font size="4">
          <i>Resposta:</i>
        </font>
      </p>
      <xsl:element name="textarea">
        <xsl:attribute name="name">
          Q<xsl:value-of select="count(preceding::questao)+1"/>
        </xsl:attribute>
        <xsl:attribute name="rows">7</xsl:attribute>
        <xsl:attribute name="cols">74</xsl:attribute>
      </xsl:element>
      <br/><br/>
    </xsl:if>
    <br/><hr color="#b0c4de"/><br/>
  </xsl:template>
```

**Notas:**

**linhas 99-133** – Template que vai gerar os campos do formulário para cada questão.

**linha 112** – Apresenta o enunciado da questão.

**linha 116** – Manda processar as alíneas.

**linhas 117-131** – Se a pergunta não tiver alíneas, então é uma pergunta de desenvolvimento; neste caso, é criada uma caixa para o aluno poder desenvolver a sua resposta.

```
  <xsl:template match="alinea">
    <tr>
      <td><li><xsl:apply-templates/></li></td>
      <td>
        <p align="right">
          <font size="-2">
            <xsl:text>V</xsl:text>
            <xsl:element name="input">
              <xsl:attribute name="type">radio</xsl:attribute>
              <xsl:attribute name="name">
                <xsl:value-of select="count(preceding::alinea)+1"/>
              </xsl:attribute>
              <xsl:attribute name="value">true</xsl:attribute>
            </xsl:element>
            <xsl:text>F</xsl:text>
            <xsl:element name="input">
              <xsl:attribute name="type">radio</xsl:attribute>
              <xsl:attribute name="name">
                <xsl:value-of select="count(preceding::alinea)+1"/>
              </xsl:attribute>
              <xsl:attribute name="value">false</xsl:attribute>
            </xsl:element>
          </font>
        </p>
      </td>
    </tr>
  </xsl:template>
```

```
161 ...
162 </xsl:stylesheet>
```

**Notas:**

**linhas 134-160** – Nesta template, é tratada cada uma das alíneas; para cada alínea são criados dois botões com os valores `Verdadeiro` e `Falso` respectivamente.

A transformação para LaTeX é muito semelhante bastando trocar as anotações geradas por outras.

Após a transformação do documento XML para HTML do exame, vamos, na próxima secção, ver como é tratada a interacção respeitante à correcção do exame e do feed-back a dar ao aluno.

# 5 Correcção online

Um formulário HTML implica interacção, neste caso com o servidor do exame. Assim, do lado servidor, terá que haver um programa que receba os dados do aluno, os processe e envie uma resposta ao aluno.

A tecnologia necessária para a implementação deste mecanismo está há muito desenvolvida e dá pelo nome de CGI (”Common Gateway Interface”). Uma CGI pode ser implementada com uma linguagem de `Scripting`: Perl, Python, C, Java, ASP, ... No nosso caso, optamos pelo Perl porque vai ser necessário processar o documento XML do exame (para ir buscar as soluções das questões e cruzá-las com as respostas do aluno) e existe um módulo Perl de nome XML::DT para processar XML que tem vindo a ser desenvolvido no Departamento de Informática [RA99].

Recorrendo a esta tecnologia foi possível montar uma arquitectura funcional com a seguinte dinâmica:

1. o docente cria o exame em XML, recorrendo a um editor estruturado ou a um simples editor de texto, e submete-o ao sistema;
2. o sistema gera a versão HTML do exame que fica disponível num determinado URL;
3. o aluno acede ao URL do exame;
4. o servidor envia-lhe o formulário correspondente;
5. o aluno utiliza o formulário para responder ao exame e submete-o ao sistema (vai ser uma CGI que vai atender este pedido);
6. o servidor recebe o exame resolvido e encaminha-o para a respetiva CGI;
7. a CGI vai cruzar a informação recebida com as soluções deixadas pelo docente no exame, realiza os cálculos necessários e envia uma resposta;
8. o aluno recebe a resposta indicando a cotação obtida (para as questões de escolha múltipla) e a resposta correcta às questões que errou.

Na secção seguinte, apresentam-se com mais detalhe as CGIs desenvolvidas.

## 5.1 Elaboração das CGI’s do Xexam

Aquando da elaboração da folha de estilo para gerar a página HTML do exame, a template que transforma o nodo `corpo` continha as seguintes linhas:

```
1 ...
2   <form action="http://localhost/cgi-bin/cgi2.cgi" method="post">
3 ...
```

Repare no atributo `action` do elemento `form` que indica ao servidor qual a CGI a executar quando o pedido é enviado, e é de notar também o elemento `input` que permite definir parâmetros a passar ao servidor.

Após este pequeno extracto de código, vamos ver como elaborar a CGI’s para a correcção automática.

### 5.1.1 Correcção do exame

Apresenta-se a seguir um extracto da CGI que faz a correcção do exame.

```
1  ...
2  my %handler=(
3      -type => {  alineas => 'SEQ',
4                  questao => 'MAP',
5                exame => 'SEQ',
6                corpo => 'SEQ',
7                xref => 'SEQ',
8             },
9       '-outputenc' => 'ISO-8859-1',
10      '-default' => sub{$c},
11      'cabecalho' => sub{""},
12      'anexo' => sub{""},
13      'alinea' => sub{ $sol{++$k}=$v{validacao};
14                       +{val => $v{validacao}, texto => $c}},# remember $v{validacao}
15      'questao' => sub{+{n=>$v{numeroQ}, %$c }},# remember $v{numeroQ}
16      'lista_seq' => sub{lista_seq($c,$v)},# remember $v{tipo}
17      'tabela' => sub{tabela($c,$v)},# remember $v{titulo}
18      'item' => sub{li(dd,$c)},
19      'linha_cod' => sub{$c},
20      'grafico' => sub{grafico($v)},# remember $v{path}
21      'figura' => sub{figura($c)},
22      'xref' => sub{xref($c)},
23      'ref' => sub{ref_($v)},# remember $v{label},$v{destino}
24      'linhas' => sub{'<tr>'.$c.'</tr>'},
25      'coluna' => sub{td($c)},
26      'codigo' => sub{$c},
27      'destaque' => sub{destaque($c,$v)},# remember $v{tipo}
28 );
29 my $tree = dt($par{exame},%handler);
30 my $tre = @{$tree}[1]; # para trabalhar apenas o corpo
```

**Notas:**

**linhas 2-28** – Nestas linhas define-se o array associativo `handler`; este array corresponde a uma associação entre o nome de um elemento e a respectiva função de transformação; neste caso, as funções de transformação calculam as estruturas de dados necessárias à correcção do exame.

**linhas 29-30** – Realizam as travessias ao documento XML aplicando as transformções definidas acima.

```
31 ...
32 print header,
33 start_html( -title=>'teste',
34             -author=>'Sao@Edgar.pt',
35             "style='margin-left: 10%; margin-right: 10%'",
36           );
37 start_form(),
38 classifica(),
39 end_form(),
40 end_html();
```

**Notas:**

**linhas 32-40** – Estas linhas estão a gerar a página de resposta; como se pode ver a função `classifica` é que vai calcular o resultado obtido.

```
sub classifica{
   my $erros = 0;      # para contar o nº de respostas erradas
   my $nresp = 0;      # para contar o nº de respostas não respondidas
   my $respC = 0;      # para contar o nº de respostas Correctas
   my $cotacao = 0;    # somar as cotações das respostas certas
   my $valAlinea = 20/$k; # valor a atribuir a cada alinea

    print br.br.
         "<table align='center'
           border='2'
         BORDERCOLOR ='b0c4de'
         width='80%'>
         <tr>
         <td BORDERCOLOR='#FFFFFF'>".br;

   for(keys %sol){
     if((($sol{$_} eq "true") && ($par{$_} eq "false"))
         or (($sol{$_} eq "false") && ($par{$_} eq "true")))
        {$erros++;
    }
    else {
       if(!defined($par{$_}))
        {$nresp++;}
       else{ $cotacao = $cotacao+$valAlinea;
             $respC++;}
    }
   }
   if(($cotacao-($valAlinea*$erros)) > 9.4){
      print h3({-align=>'center'},
               font({-color=>'green'},"Aprovado!!")),
        br,dd.dd.dd.font({-size=>'4'},i("Resultado: ")),
        font({-size=>'3'},
        ' ',$cotacao-$valAlinea*$erros," valores");
    }
   else {
      if(($cotacao-($valAlinea*$erros)) < 0){
        print h3({-align=>'center'},
             font({-color=>'red'},"Reprovado!!")),
              br,dd.dd.dd.font({-size=>'4'},i("Resultado: ")),
          font({-size=>'3'},' ',"0 valores");
          }
      else{ print h3({-align=>'center'},
                     font({-color=>'red'},"Reprovado!!")),
                   font({-size=>'4'},br,dd.dd.dd.i("Resultado: ")),
          font({-size=>'3'},
          $cotacao-$valAlinea*$erros,' '," valores");
          }
     }
  print br,font({-size=>'3'},
    dd.dd.dd.dd.$respC,' '," respostas correctas",br,
        dd.dd.dd.dd.$erros,' '," respostas erradas",br,
    dd.dd.dd.dd.$nresp,' '," respostas não respondidas"),br;

  print "</table></tr></td>";
  print br,br;
  mkform($tre,\%sol,\%par);
 }
 ...
```

Este último excerto de código mostra a função classifica que faz uma travessia às estruturas de dados criadas na travessia do documento XML do exame, e calcula o resultado final.

Este artigo termina com a próxima secção onde se tiram algumas conclusões e se faz uma síntese do que foi feito e do que falta fazer.

# 6 Conclusão

Ao longo deste artigo nunca foram feitas comparações com soluções já desenvolvidas. Isto deve-se ao facto de, na comunidade de E-Learning, não existirem standards, existem sim conjuntos de recomendações que as entidades poderão seguir ou não. Apesar disso, existem dois grandes grupos que integram, cada um, algumas centenas de instituições de ensino: SCORM e IMS.

No que à avaliação diz respeito, estes grupos publicaram um DTD de uma linguagem que, segundo eles, deverá ser utilizada na produção de exames. No entanto, a linguagem definida nesse DTD é bastante complexa (é o que normalmente acontece quando se produzem DTDs para grandes comunidades). Além disso, também não disponibilizam ferramentas para colocar o sistema a funcionar.

Foi por estas razões, que no contexto do projecto aqui retratado se decidiu desenvolver uma pequena linguagem para testar algumas ideias.

O protótipo desenvolvido demonstrou a sua viabilidade e utilidade. Num futuro próximo, estará a ser utilizado no apoio à auto-avaliação dos alunos.

Há, no entanto, muitas coisas a melhorar e a implementar. Tais como:

- Aceitar mais tipos de exame: com perguntas aleatórias, com questões de verdadeiro e falso, progressivos, com tempo limite, ...
- Adicionar um módulo para autenticação do aluno.
- Criar uma base de dados para registo das interacções com os alunos.
- Desenvolver um módulo para correcção automática para perguntas de desenvolvimento de cálculo ou de programação.

# Referências


[HM01] Elliotte Rusty Harold and W. Scott Means. *XML in a Nutshell*. O'Reilly, 1ª edição edition, Janeiro 2001.

[RA99] J.C. Ramalho and J.J. Almeida. Xml::dt - a perl down translation module. In *XML Europe'99*, Granada - Espanha, Abril 1999.

[RH95] José Carlos Ramalho and Pedro Rangel Henriques. Uma experiência de utilização da internet na gestão pedagógica e sua formalização. In *Conferência Nacional WWW*. Universidade do Minho, 1995.

[RH02] José Carlos Ramalho and Pedro Rangel Henriques. *XML e XSL: da teoria à prática*. 2002.

[Wil01] Heather Williamson. *XML: the complete reference*. Osborne/McGraw-Hill, 2001.