6
GLOSSÁRIO

Arquivo

Conjunto de dados digitalizados que pode ser gravado em um dispositivo de armazenamento e tratado como ente único. Arquivos podem conter representações de documentos, figuras estáticas ou em movimentos, sons e quaisquer outros elementos capazes de serem digitalizados.

Catálogo de metadados

Repositório de elementos de metadados que descrevem determinado conjunto de dados.

Composição

Tipo de agrupamento de dados que pressupõe a interdependência entre eles, de modo que cada um não têm significado isolado.

Dado

Composição identificável de dados digitalizados, percebidos como um único ente, que pode ser gravado em um recipiente.

Dado geográfico

Dado associado com coordenadas geográficas relacionadas com ele.

Obs.: Um dado geográfico pode ser um grupo de dados menor que, limitado por alguma restrição como extensão espacial ou tipo de referencial, está fisicamente contido em um dado geográfico maior. Teoricamente, um dado geográfico pode ser tão pequeno quanto um único referencial ou atributo de referencial contido em um dado geográfico maior. Uma cópia de um mapa e um gráfico podem ser dados geográficos.

Descrição de um dado

Conjunto de elementos de metadados que descrevem o significado, natureza, origem, validade, e outras características relevantes para seleção e acesso ao dado.

Dicionário

Repositório de objetos que dispõe de um protocolo para consulta eletrônica.

Dicionário geográfico

Dicionário de objetos geográficos. Fornece uma lista de objetos como as que existem ao final de um Atlas.

Elemento de metadado

Unidade discreta de metadado. Pode utilizar-se um tesauro ou um dicionário de objetos como domínio para um elemento de metadado.

Esquema de metadados

Representação conceitual da estrutura computacional utilizada para armazenamento dos elementos de metadados.

Metadado

Dados utilizados para descrever outro dado.

Objeto

Algo que constitui uma referência a um dado. Algo que está relacionado a ele. Uma abstração de um fenômeno do mundo real.

Objeto geográfico

Um objeto geográfico é um objeto associado a uma localização sobre a superfície terrestre. Um objeto geográfico representa um lugar, tal como a cidade do Rio de Janeiro.

Recipiente

Unidade de armazenamento de um dado.

Obs.: As unidades de armazenamento usuais são arquivos, registros, atributos, tabelas, etc.

Repositório

Dispositivo para armazenamento e/ou proteção de dados e/ou suas descrições.

Tesauro

Lista de sinônimos classificada. Repertório alfabético de termos utilizados em indexação e na classificação de documentos. Vocabulário de um ramo do saber que descreve sem ambigüidade os conceitos a ele atinentes.

7
REFERÊNCIAS

[1] Clementini, E.; di Felice, P.; van Oosterom, P., 1993. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In: Abel, D.; Ooi, B. C., eds., SSD '93: Lecture Notes in Computer Science, v. 692: New York, NY, Springer-Verlag, p. 277-295.

[2] Klien, E.; Lutz, M., 2005. The Role of Spatial Relations in Automating the Semantic Annotation of Geodata. In Conf. on Spatial Information Theory.

[3] Souza, L.A. et al., 2005. The Role of Gazetteers in Geographic Knowledge Discovery on the Web. In Proc. 3rd Latin American Web Congress. Buenos Aires, Argentina (Oct. 2005).

[4] Hollink, L., Schreiber, G., Wielemaker, J. and Wielinga, B. (2003) "Semantic Annotation of Image Collections". In S. Handschuh, M. Koivunen, R. Dieng and S. Staab (eds.): Knowledge Capture 2003 -- Proceedings Knowledge Markup and Semantic Annotation Workshop, October 2003.

[5] Hiramatsu, K. and Reitsma, F. (2004). "GeoReferencing the Semantic Web: ontology based markup of geographically referenced information", Joint EuroSDR / EuroGeographics Workshop.

[6] Alexandria Digital Library Gazetteer. (1999). Santa Barbara CA: Map and Imagery Lab, Davidson Library, University of California, Santa Barbara. Copyright UC Regents.

[7] Hill, L.; Frew, J.; Zheng, Q., 1999. Geographic Names: The Implementation of a Gazetteer in a Georeferenced Digital Library. In: D-Lib, January-1999.

[8] Nebert, D., 2002. Catalog Services Specification, Version 1.1.1, OpenGIS© Implementation Specification, Open GIS Consortium, Inc.

[9] ISO-2788, 1986. Documentation -- Guidelines for the development of monolingual thesauri, International Standard ISO-2788, Second edition -- 1986-11-15

[10] ISO-19115, 2003. ISO 19115 Geographic information – Metadata. Draft International Standard.

[11] Senkler, K.; Voges, U.; Remke, A., 2004. An ISO 19115/19119 Profile for OGC Catalogue Services CSW 2.0. In 10th EC GI & GIS Workshop, ESDI State of the Art, Warsaw, Poland, 23-25 June 2004.

[12] FGDC, 2002. Federal Geographic Data Committee.

[13] GNIS, 2005. Geographic Names Information System, U.S. Department of the Interior, U.S. Geological Survey, Reston, USA.

[14] Embrapa, 2004. Embrapa Monitoramento por Satélite. Brazil seen from Space.

[15] Gamma, Erich; Richard Helm; Ralph Jonson; John Vlissides (1995). Design patterns – elements of reusable object-oriented software.

[16] Pree, W. (1994). Meta patterns - a means for capturing the essentials of reusable object-oriented design. in M. Tokoro and R. Pareschi (eds), Springer-Verlag, proceedings of the ECOOP, Bologna, Italy: 150-162

[17] Pressman, Roger S. (1992). Software Engineering: A Practitioner's Approach

[18] Percivall, G. (Ed) 2003. OpenGIS® Reference Model. Document number OGC 03-040 Version: 0.1.3. Open Geospatial Consortium, Inc.

[19] Percivall, G. (Ed) (2002) The OpenGIS Abstract Specification - Topic 12: OpenGIS Service Architecture Version 4.3. OpenGIS Project Document Number 02-112. Open GIS Consortium.

[20] Whiteside, A. (Ed) (2005) OpenGIS® web services architecture description, Version: 0.1.0. OpenGIS® Best Practices, OGC 05-042r2. Open Geospatial Consortium Inc.

[21] TGN (2004) Thesaurus of Geographical Names. The Getty Foundation.

[22] Fitzke, J.; Atkinson, R. (2006) Gazetteer Service Profile of the Web Feature Service Implementation Specification, Version 0.9.1. OGC® Implementation Specification, OGC 05-035r1. Open Geospatial Consortium Inc.

8
APÊNDICE A – Protocolo de Serviço do ADL Gazetteer (esquemas XML)

Esquema Principal

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:gaz="http://www.alexandria.ucsb.edu/gazetteer"
  targetNamespace="http://www.alexandria.ucsb.edu/gazetteer"
  elementFormDefault="qualified">
  <annotation>
    <appinfo>
      <header xmlns=""> $Header:
       /export/home/gjanee/gazetteer/RCS/gazetteer-service.xsd,v
       1.3 2003/09/17 20:48:30 gjanee Exp $
      </header>
      <log xmlns="">
        $Log: gazetteer-service.xsd,v $ Revision 1.3 2003/09/17
        20:48:30 gjanee Minor change to accommodate a rename.
        Also, slightly recoded to take advantage of the new
        'report-format-type' element type. Revision 1.2
        2003/09/17 08:16:46 gjanee Removed all references to the
        defunct 'add-entry', 'relate-entries', and 'remove-entry'
        services. Changed how success and failure are indicated.
        The &lt;error&gt; element is no longer required but
        nillable; instead, it is now optional and its presence or
        absence indicates failure or success. Revision 1.1
        2001/07/08 04:39:56 gjanee Initial revision
      </log>
    </appinfo>
    <documentation>
      <author xmlns="">
        <name>Greg Jan&#x00E9;e</name>
        <affiliation>Alexandria Digital Library Project
        </affiliation>
        <email-address>gjanee@alexandria.ucsb.edu</email-address>
      </author>
    </documentation>
  </annotation>
  <include schemaLocation="gazetteer-capabilities.xsd" />
  <include schemaLocation="gazetteer-query.xsd" />
  <include schemaLocation="gazetteer-standard-report.xsd" />
  <include schemaLocation="gazetteer-types.xsd" />
  <element name="gazetteer-service">
    <complexType>
      <choice>
        <element ref="gaz:get-capabilities-request" />
        <element ref="gaz:get-capabilities-response" />
        <element ref="gaz:query-request" />
        <element ref="gaz:query-response" />
        <element ref="gaz:download-request" />
```

```
      <element ref="gaz:download-response" />
    </choice>
    <attribute name="version" type="string" use="required" />
  </complexType>
</element>
<element name="get-capabilities-request">
  <complexType />
</element>
<element name="get-capabilities-response">
  <complexType>
    <sequence>
      <element ref="gaz:gazetteer-capabilities" minOccurs="0" />
      <element ref="gaz:error" minOccurs="0" />
    </sequence>
  </complexType>
</element>
<element name="query-request">
  <complexType>
    <sequence>
      <element ref="gaz:gazetteer-query" />
      <element name="report-format" type="gaz:report-format-
      type" />
      <element name="geometry-language" type="anyURI"minOccurs=
      "0"/>
    </sequence>
  </complexType>
</element>
<element name="query-response">
  <complexType>
    <sequence>
      <choice minOccurs="0">
        <element name="standard-reports">
          <complexType>
            <sequence>
              <element ref="gaz:gazetteer-standard-report"
                minOccurs="0" maxOccurs="unbounded" />
            </sequence>
          </complexType>
        </element>
        <element name="extended-reports">
          <complexType>
            <sequence>
              <any processContents="lax" minOccurs="0"
                maxOccurs="unbounded" />
            </sequence>
          </complexType>
        </element>
      </choice>
      <element ref="gaz:error" minOccurs="0" />
    </sequence>
  </complexType>
</element>
<element name="download-request">
  <complexType>
    <sequence>
      <element name="report-format" type="gaz:report-format-
      type" />
      <element name="geometry-language" type="anyURI"minOccurs=
      "0"/>
    </sequence>
  </complexType>
```

```
  </element>
  <element name="download-response">
    <complexType>
      <sequence>
        <choice minOccurs="0">
          <element name="standard-reports">
            <complexType>
              <sequence>
                <element ref="gaz:gazetteer-standard-report"
                  minOccurs="0" maxOccurs="unbounded" />
              </sequence>
            </complexType>
          </element>
          <element name="extended-reports">
            <complexType>
              <sequence>
                <any processContents="lax" minOccurs="0"
                  maxOccurs="unbounded" />
              </sequence>
            </complexType>
          </element>
        </choice>
        <element ref="gaz:error" minOccurs="0" />
      </sequence>
    </complexType>
  </element>
  <element name="error">
    <complexType>
      <sequence>
        <element name="code" type="string" minOccurs="0" />
        <element name="description" type="string" minOccurs="0" />
      </sequence>
    </complexType>
  </element>
</schema>
```

Esquema para consultas

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:gaz="http://www.alexandria.ucsb.edu/gazetteer"
  xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.alexandria.ucsb.edu/gazetteer"
  elementFormDefault="qualified">
  <annotation>
    <appinfo>
      <header xmlns="">
        $Header: /export/home/gjanee/gazetteer/RCS/gazetteer-
        query.xsd,v
        1.4 2003/10/07 21:39:26 gjanee Exp $
      </header>
      <log xmlns="">
        $Log: gazetteer-query.xsd,v $ Revision 1.4 2003/10/07
        21:39:26 gjanee Added subelement &lt;code-query&gt;.
        Revision 1.3 2003/09/17 20:45:11 gjanee Minor change to
        accommodate a rename. Revision 1.2 2003/09/17 08:00:59
        gjanee Added subelement &lt;place-status-query&gt;. In
        subelement &lt;relationship-query&gt;, renamed attributes
        'relationship' and 'identifier' to 'relation' and
```

```
        'target-identifier', respectively. Revision 1.1
        2001/06/21 20:37:16 gjanee Initial revision
      </log>
    </appinfo>
    <documentation>
      <author xmlns="">
        <name>Greg Jan&#x00E9;e</name>
        <affiliation>Alexandria Digital Library Project
        </affiliation>
        <email-address>gjanee@alexandria.ucsb.edu</email-address>
      </author>
    </documentation>
  </annotation>
  <include schemaLocation="gazetteer-types.xsd" />
  <import namespace="http://www.opengis.net/gml"
    schemaLocation="geometry.xsd" />
  <element name="gazetteer-query">
    <complexType>
      <sequence>
        <group ref="gaz:query" />
      </sequence>
    </complexType>
  </element>
  <group name="query">
    <choice>
      <element ref="gaz:identifier-query" />
      <element ref="gaz:code-query" />
      <element ref="gaz:place-status-query" />
      <element ref="gaz:name-query" />
      <element ref="gaz:footprint-query" />
      <element ref="gaz:class-query" />
      <element ref="gaz:relationship-query" />
      <element ref="gaz:and" />
      <element ref="gaz:or" />
      <element ref="gaz:and-not" />
    </choice>
  </group>
  <element name="identifier-query">
    <complexType>
      <attribute name="identifier" type="string" use="required" />
    </complexType>
  </element>
  <element name="code-query">
    <complexType>
      <attribute name="scheme" type="string" />
      <attribute name="code" type="string" use="required" />
    </complexType>
  </element>
  <element name="place-status-query">
    <complexType>
      <attribute name="status" type="gaz:status-type"
       use="required"/>
    </complexType>
  </element>
  <element name="name-query">
    <complexType>
      <attribute name="operator" use="required">
        <simpleType>
          <restriction base="string">
            <enumeration value="contains-all-words" />
            <enumeration value="contains-any-words" />
```

```
          <enumeration value="contains-phrase" />
          <enumeration value="equals" />
          <enumeration value="matches-pattern" />
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="text" type="string" use="required" />
  </complexType>
</element>
<element name="footprint-query">
  <complexType>
    <choice>
      <element ref="gml:Box" />
      <element ref="gml:Polygon" />
      <element name="identifier" type="string" />
      <element name="other-region">
        <complexType>
          <sequence>
            <any processContents="lax" />
          </sequence>
        </complexType>
      </element>
    </choice>
    <attribute name="operator" use="required">
      <simpleType>
        <restriction base="string">
          <enumeration value="contains" />
          <enumeration value="overlaps" />
          <enumeration value="within" />
        </restriction>
      </simpleType>
    </attribute>
  </complexType>
</element>
<element name="class-query">
  <complexType>
    <attribute name="thesaurus" type="string" use="required" />
    <attribute name="term" type="string" use="required" />
  </complexType>
</element>
<element name="relationship-query">
  <complexType>
    <attribute name="relation" type="string" use="required" />
          <attribute name="target-identifier" type="string"
           use= "required"/>
  </complexType>
</element>
<element name="and">
  <complexType>
    <sequence>
      <group ref="gaz:query" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
<element name="or">
  <complexType>
    <sequence>
      <group ref="gaz:query" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
```

```
  <element name="and-not">
    <complexType>
      <sequence>
        <group ref="gaz:query" minOccurs="2" maxOccurs="2" />
      </sequence>
    </complexType>
  </element>
</schema>
```

Esquema para a resposta da consulta

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:gaz="http://www.alexandria.ucsb.edu/gazetteer"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  targetNamespace="http://www.alexandria.ucsb.edu/gazetteer"
  elementFormDefault="qualified">
  <annotation>
    <appinfo>
      <header xmlns=""> $Header:
       /export/home/gjanee/gazetteer/RCS/gazetteer-standard-
       report.xsd,v 1.4 2003/10/07 21:48:49 gjanee Exp $ </header>
      <log xmlns="">
        $Log: gazetteer-standard-report.xsd,v $ Revision 1.4
        2003/10/07 21:48:49 gjanee Added subelement
        &lt;codes&gt;. Revision 1.3 2003/09/17 20:44:32 gjanee
        Minor change to accommodate a rename. Revision 1.2
        2003/09/17 07:46:42 gjanee Replaced the 'historical'
        attribute with the new tri-valued 'status' attribute.
        Added subelements &lt;place-status&gt; and &lt;display-
        name&gt;. In the &lt;relationship&gt; subelement, renamed
        the 'name' and 'identifier' attributes to 'relation' and
        'target-identifier', respectively, and added a 'target-
        name' attribute. To support virtual relationships, the
        'target-identifier' attribute is now optional. Revision
        1.1 2001/07/05 13:43:35 gjanee Initial revision
      </log>
    </appinfo>
    <documentation>
      <author xmlns="">
        <name>Greg Jan&#x00E9;e</name>
        <affiliation>Alexandria Digital Library Project
        </affiliation>
        <email-address>gjanee@alexandria.ucsb.edu</email-address>
      </author>
    </documentation>
  </annotation>
  <include schemaLocation="gazetteer-types.xsd" />
  <import namespace="http://www.opengis.net/gml"
    schemaLocation="geometry.xsd" />
  <import namespace="http://www.w3.org/1999/xlink"
    schemaLocation="xlinks.xsd" />
  <attributeGroup name="qualifiers">
    <attribute name="primary" type="boolean" default="false" />
    <attribute name="status" type="gaz:status-
     type"default="current"/>
  </attributeGroup>
  <element name="gazetteer-standard-report">
```

```
<complexType>
  <sequence>
    <element name="identifier" type="string" />
    <element name="codes" minOccurs="0">
      <complexType>
        <sequence>
          <element name="code" minOccurs="0"
            maxOccurs="unbounded">
            <complexType>
              <simpleContent>
                <extension base="string">
                  <attribute name="scheme" type="string"
                    use="required" />
                </extension>
              </simpleContent>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
    <element name="place-status" type="gaz:status-type" />
    <element name="display-name" type="string" />
    <element name="names">
      <complexType>
        <sequence>
          <element name="name" maxOccurs="unbounded">
            <complexType>
              <simpleContent>
                <extension base="string">
                  <attributeGroup ref="gaz:qualifiers" />
                </extension>
              </simpleContent>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
    <element name="bounding-box" type="gml:BoxType" />
    <element name="footprints">
      <complexType>
        <choice maxOccurs="unbounded">
          <element name="footprint">
            <complexType>
              <choice>
                <element ref="gml:_Geometry" />
                <element ref="gml:Box" />
                <element name="other-footprint">
                  <complexType>
                    <sequence>
                      <any processContents="lax" />
                    </sequence>
                  </complexType>
                </element>
              </choice>
              <attributeGroup ref="gaz:qualifiers" />
            </complexType>
          </element>
          <element name="footprint-reference">
            <complexType>
              <attributeGroup ref="xlink:locatorLink" />
              <attribute name="geometry-type">
```

```
                    <simpleType>
                      <restriction base="string">
                        <enumeration value="Box" />
                        <enumeration value="Point" />
                        <enumeration value="LineString" />
                        <enumeration value="Polygon" />
                        <enumeration value="MultiPoint" />
                        <enumeration value="MultiLineString" />
                        <enumeration value="MultiPolygon" />
                        <enumeration value="other" />
                      </restriction>
                    </simpleType>
                  </attribute>
                  <attribute name="num-points"
                   type="positiveInteger "/>
                  <attributeGroup ref="gaz:qualifiers" />
                </complexType>
              </element>
            </choice>
          </complexType>
        </element>
        <element name="classes" minOccurs="0">
          <complexType>
            <sequence>
              <element name="class" minOccurs="0"
                maxOccurs="unbounded">
                <complexType>
                  <simpleContent>
                    <extension base="string">
                      <attribute name="thesaurus" type="string"
                        use="required" />
                      <attributeGroup ref="gaz:qualifiers" />
                    </extension>
                  </simpleContent>
                </complexType>
              </element>
            </sequence>
          </complexType>
        </element>
        <element name="relationships" minOccurs="0">
          <complexType>
            <sequence>
              <element name="relationship" minOccurs="0"
                maxOccurs="unbounded">
                <complexType>
                  <attribute name="relation" type="string"
                    use="required" />
                  <attribute name="target-name" type="string"
                    use="required" />
                  <attribute name="target-identifier" type=
                   "string" />
                </complexType>
              </element>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```

9
APÊNDICE B – Casos de teste

Testes caixa branca

O teste caixa branca é um método de projeto de casos de teste que usa a estrutura de controle do projeto procedural para derivar casos de teste.

Usando métodos de teste de caixa branca, pode-se derivar os casos de teste que: garantam que todos os caminhos independentes dentro de um módulo tenham sido exercitados pelo menos uma vez; exercitem todas as decisões lógicas para valores falsos ou verdadeiros; executem todos os laços em suas fronteiras e dentro de seus limites operacionais; e exercitem as estruturas de dados internas para garantir a sua validade.

Teste caixa preta

Os métodos de teste de caixa preta concentram-se nos requisitos funcionais do software. Ou seja, esse teste possibilita derivar conjuntos de condições de entrada que exercitem completamente todos os requisitos funcionais para um programa. O teste de caixa preta procura descobrir erros nas seguintes categorias: funções incorretas ou ausentes; erros de interface; erros nas estruturas de dados ou no acesso a bancos de dados externos; erros de desempenho; e erros de inicialização e término.

Estratégias de teste

O processo de engenharia de software pode ser visto como uma espiral [17], conforme ilustra a Figura 30. Inicialmente, a engenharia de sistemas define o papel do software e leva à análise de requisitos de software, na qual o domínio de informação, função, comportamento, desempenho, imposições e critérios de validação para o software são estabelecidos. Movimentando-se para o dentro ao longo da espiral, chegamos ao projeto e, finalmente, à codificação. Para

desenvolver um software de computador, seguimos a espiral ao longo de linhas aerodinâmicas que diminuem o nível de abstração em cada volta.

Uma estratégia de teste de software também pode ser imaginada ao se movimentar para fora na espiral. O teste unitário inicia-se no vértice da espiral e concentra-se em cada unidade do software, de acordo, com o que é implementado no código-fonte. A atividade de teste progride ao se movimentar para fora, ao longo da espiral, para o teste de integração, onde a atenção encontra-se no projeto e na construção da arquitetura de software. Avançando-se na curva para fora na espiral, encontramos o teste de validação, nos quais os requisitos estabelecidos como parte da análise de requisitos de software são validados em relação ao software que foi construído. Finalmente, chegamos ao teste de sistema, no qual o software e outros elementos do sistema são testados como um todo. Para testar o software de computador, deslocamo-nos pela espiral ao longo de linhas aerodinâmicas que ampliam o campo de ação de teste em cada volta.

Considerando-se o processo de um ponto de vista procedimental, a atividade de teste dentro do contexto da engenharia de software é, de fato, uma série de três passos que é implementada sequencialmente. Inicialmente, os testes focalizam cada módulo individualmente, garantindo que ele funcione adequadamente como uma unidade. Por isso, o nome de teste de unidade. O teste de unidade faz muito uso das técnicas de teste de caixa branca, exercitando caminhos específicos da estrutura de controle de um módulo, a fim de garantir uma completa cobertura e máxima detecção de erros. Em seguida, os módulos devem ser montados ou integrados para formarem um pacote de software. O teste de integração cuida das questões associadas aos duplos problemas de verificação e construção de programas. As técnicas de projeto de casos de teste de caixa preta são as mais encontradas durante a integração, não obstante uma quantidade limitada de teste de caixa branca possa ser usada para garantir a cobertura de caminhos importantes de controle.

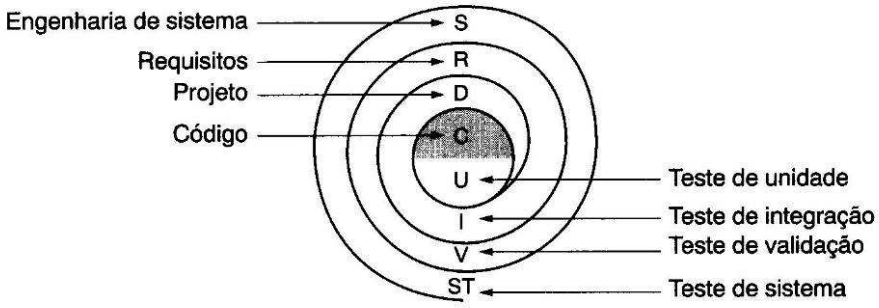

Figura 30 - Estratégia de teste

Depois que o software foi integrado (construído), um conjunto de testes de alto nível é realizado. Os critérios de validação (estabelecidos durante a análise de requisitos) devem ser testados. O teste de validação oferece a garantia final de que o software atende a todas as exigências funcionais, comportamentais e de desempenho. As técnicas de caixa preta são usadas exclusivamente durante essa validação.

A última etapa de testes de alto nível extrapola a fronteira da engenharia de software e situa-se no contexto da engenharia de sistemas computadorizados. O software, uma vez validado, deve ser combinado com outros elementos do sistema (por exemplo, hardware, pessoas, bancos de dados). O teste de sistema verifica se todos os elementos combinam-se adequadamente e se a função/desempenho global do sistema é conseguida.

Teste de Unidade

O teste de unidade concentra-se no esforço de verificação da menor unidade de projeto de software – o módulo. Usando a descrição do projeto detalhado como guia, caminhos de controle importantes são testados para descobrir-se erros dentro das fronteiras do módulo. A complexidade relativa dos testes e os erros detectados como resultado deles são limitados pelo campo de ação restrito estabelecido para o teste de unidade. O teste de unidade baseia-se sempre na caixa branca, e esse passo pode ser realizado em paralelo para múltiplos módulos.

Nesse tipo de teste a interface com o módulo é testada para ter a garantia de que as informações fluem para dentro e para fora da unidade de programa que se encontra sob teste. A estrutura de dados local é examinada para ter a

garantia de que os dados armazenados temporariamente mantêm sua integridade durante todos os passos de execução de um algoritmo. As condições de limite são testadas para ter a garantia de que o módulo opera adequadamente nos limites estabelecidos para demarcarem ou restringirem o processamento.

Todos os caminhos independentes (caminhos básicos) através da estrutura de controle são exercitados para ter a garantia de que todas as instruções de um módulo foram executadas pelo menos uma vez. E, finalmente, todos os caminhos de tratamento de erros são testados.

Uma vez que um módulo não é um programa individual, um software *driver* e/ou *stub* deve ser desenvolvido para cada unidade de teste. Um *driver* nada mais é do que um programa que aceita dados de caso de teste e que passa tais dados para o módulo a ser testado. Os *stubs* servem para substituir módulos que estejam subordinados ao módulo a ser testado. Um *stub* utiliza a interface do módulo subordinado, pode fazer manipulação de dados mínima, imprime verificação de entrada e retorna.

Teste de integração

O teste de integração é uma técnica sistemática para a construção da estrutura de programa, realizando-se, ao mesmo tempo, testes para descobrir erros associados a interfaces. O objetivo é, a partir dos módulos testados no nível de unidade, construir a estrutura de programa que foi determinada pelo projeto.

Integração *top-down* é uma abordagem incremental à construção da estrutura de programa. Os módulos são integrados movimentando-se de cima para baixo através da hierarquia de controle, iniciando-se do módulo de controle principal. Os módulos subordinados ao módulo de controle principal são incorporados à estrutura de duas formas: primeiro em profundidade ou primeiro em largura. Na primeira forma os módulos são incorporados de maneira análoga ao caminhamento primeiro em profundidade em um grafo. Na segunda assemelha-se ao caminhamento em amplitude em um grafo.

O teste de *bottom-up*, como seu nome indica, inicia a construção e os testes com módulos atômicos (módulos localizados nos níveis mais baixos da estrutura de programa). Uma vez que os módulos são integrados de baixo para cima, o processamento exigido para os módulos subordinados em determinado nível está sempre disponível, e a necessidade de *stubs* é eliminada.

Teste de validação

Ao término da atividade de teste de integração, o software está completamente montado como um pacote, erros de interface foram descobertos e corrigidos e uma série final de testes, os testes de validação, de software pode iniciar-se. A validação pode ser definida de muitas maneiras, mas uma definição simples é que a validação é bem-sucedida quando o software funciona de uma maneira razoavelmente esperada pelo cliente. São definidas expectativas razoáveis na Especificação de Requisitos de Software que descreve todos os atributos do software visíveis ao usuário.

Teste de Sistema

O teste de sistema é, na verdade, uma série de diferentes testes, cujo propósito primordial é pôr completamente à prova o sistema baseado em computador. Não obstante cada teste tenha uma finalidade diferente, todo o trabalho deve verificar se todos os elementos do sistema foram adequadamente integrados e realizam as funções atribuídas.

Critérios de teste utilizados

Foram realizadas três categorias de teste no GeoCatalog: testes unitários, de integração e de validação.

Na primeira etapa de testes, os módulos que implementam as regras de negócio de recuperação de metadados da imagem e anotações dos dicionários foram testados combinando-se as estratégias de teste unitário caixa branca e teste integrado *bottom-up*. Foi utilizado nessa abordagem o *framework* JUnit como mecanismo de automação da execução dos casos de teste.

Em seguida, testes de validação, manuais, foram realizados no conjunto completo dos módulos da aplicação para colocar à prova os requisitos funcionais especificados nos casos de uso de usuário.

Na primeira etapa, casos de teste exercitando comandos de decisão e repetição, valores limite e tratamento de exceções, foram sendo aplicados desde os módulos mais atômicos (sem dependências externas) até os módulos *FileSystemDirectory* e *ADLGazeteer* que, em última análise, são os responsáveis por toda a recuperação de informações do software. Não foram utilizados *stubs*, uma vez que módulos hierarquicamente superiores na hierarquia de controle utilizavam módulos já testados subordinados a ele. Os *drivers* foram implementados estendendo-se a classe *TestCase* oferecida pelo JUnit. Para cada módulo (classe) testado há um *TestCase* associado contendo vários procedimentos de teste.

Para os testes de validação foram criados casos de teste que exercitassem cada um dos casos de uso de usuário e seus respectivos fluxos alternativos.

A seguir são apresentadas as descrições dos casos de teste unitário e de validação.

Casos de teste unitários

Caso de teste:	OOISO19115GeoImageTest.java
Propósito:	Testar a corretude dos métodos da classe OOISO19115GeoImage
Descrição	**Resultados Esperados**
testAggregateDataset: criar dois Dataset's um para metadados de uma imagem e outro para features. Atribuir valores para os metadados da imagem. Adicionar duas features. Executar o método *aggregateDataset(Dataset dataset)* que irá agregar as features às imagens.	Metadados da imagem agregados com as objetos

Caso de teste:	XMLParserTest.java	
Propósito:	Testar a corretude do método *parse( )* da classe XMLParser	
Descrição		**Resultados Esperados**
testParseWithValidFile: executar o método *parse(InputSource source)* com o parâmetro source obtido a partir de um arquivo de descrição de imagem válido		Arquivo lido e vetor MetadataElement[ ] criado com sucesso
testParseWithInvalidContent: executar o método *parse(InputSource source)* com o parâmetro source obtido a partir de um arquivo de descrição de imagem com o conteúdo inválido (sem o elemento XML identifier)		Lançar InvalidDatasetDescException porque o vetor de elementos não é coerente com uma GeoImage
testParseWithInvalidFile: executar o método *parse(InputSource source)* com o parâmetro source obtido a partir de um arquivo de descrição de imagem inválido (arquivo inexistente)		Lançar IOException porque ocorrerá erro de acesso a disco
testParseWithNullFile: executar o método *parse(InputSource source)* com o parâmetro source obtido a partir de um nome de arquivo igual a null		Lançar IOException porque ocorrerá erro de acesso a disco

Caso de teste:	ISO19115ImageBuilderTest.java	
Propósito:	Testar a corretude do método build( ) da classe ISO19115ImageBuilder	
Descrição		**Resultados Esperados**

testBuildWithInvalidCoordinate: com um vetor MetadataElement[ ] contendo coordenadas inválidas de uma imagem executar o método *build(MetadataElement[ ] elements)*	Lançar NumberFormatException porque irá ocorrer erro na conversão de String para Real da coordenada
testBuildWithNullElements: executar o método *build(MetadataElement[ ] elements)* com parâmetro null	Lançar InvalidDatasetDescException porque o vetor de elementos não é coerente com uma GeoImage
testBuildWithNullValues: exeutar o método *build(MetadataElement[ ] elements)* com um vetor de elementos correspondente aos elementos encontrados em um arquivo de descrição de uma imagem, porém com os valores de cada tag XM igual a null	Lançar NumberFormatException porque irá ocorrer erro na conversão de String para Real da coordenada
testBuildWithValidElements: executar o método *build(MetadataElement[ ] elements)* com um vetor de elementos válido	GeoImage construída com sucesso

Caso de teste:	ISO19115ADLFeaturesBuilderTest.java
Propósito:	Testar a corretude do método build( ) da classe ISO19115ADLFeaturesBuilder
Descrição	**Resultados Esperados**
testBuildWithNullElements: executar o método *build(MetadataElements[ ] elements)* com valor null	Lançar InvalidDatasetDescException

testBuildWithValidElements: com um vetor de elementos válido executar *build(MetadataElements[ ] elements)*	OOISO19115GeoImage construído com sucesso

Caso de teste:	FileSystemDirectoryTest.java
Propósito:	Testar a corretude dos métodos ListURI( ) e getContent( ) da classe FileSystemDirectory

Descrição	**Resultados Esperados**
testListUriInvalidDirectory: configurar a classe FileSystemDirectory com um nome de diretório inacessível e executar o método *listURI( )*	Lançar IOException porque ocorrerá erro de acesso a disco
testListUriNullDirectory: executar *listURI( )* com parâmetro null	Lançar IOException porque ocorrerá erro de acesso a disco
testGetContentInvalidURL: com FileSystemDirectory configurado com um diretório inválido executar o método *getContent(String uri)*	Lançar IOException porque ocorrerá erro de acesso a disco
testGetContentNullURI: com FileSystemDirectory configurado com um diretório válido executar o método *getContent(String uri)* com parâmetro null	Lançar IOException porque ocorrerá erro de acesso a disco

Caso de teste:	ADLGazeteerTest.java
Propósito:	Testar a corretude do método getContent( ) da classe ADLGazeteerService

Descrição	**Resultados Esperados**
testGetContentValid: com uma OOISO19115GeoImage contendo coordenadas válidas e iguais a (-43,26; -22,891) e (-43,203; -22,932), executar o método *getContent(Dataset dataset)* de ADLGazetteerService	Retornar uma GeoImage contendo 17 objetos.

testGetContentEmptyBoundingbox: com um OOISO19115GeoImage contendo coordenadas determinando uma região vazia (um ponto) executar o método *getContent(Dataset dataset)*	Retornar null
testGetContentInvalidBoundingbox: com uma OOISO19115GeoImage contendo coordenadas inválidas (valor > 180 ou menor do que -180) executar o método *getContent(Dataset dadaset)*	Lançar IOException porque o dicionário não retornará nenhuma resposta.
testRegisterObjectSrc: com valor null executar o método *getContent(Dataset dataset)*	Retornar null
testGetContentInvalidURL: com uma OOISI19115GeoImage contendo coordenadas válidas executar o método *getContent(Dataset dataset)* de um ADLGazetteerService configurado com uma URL inválida	Lançar IOException porque o dicionário não retornará nenhuma resposta

Casos de teste de validação

Caso de teste:	Criar área de trabalho
Propósito:	Testar a corretude do Caso de Uso Criar Área de Trabalho
Descrição	**Resultados Esperados**
Testa operação de criação de nova área de trabalho informando-se nome de arquivo inválido (arquivo já aberto, diretório inexistente ou acesso negado).	sistema exibe mensagem sinalizando o erro
Testa operação de criação de nova área de trabalho informando-se nome de arquivo válido.	nova área de trabalho é criada

Caso de teste:	Abrir Área de Trabalho
Propósito:	Testar a corretude do Caso de Uso Abrir Área de Trabalho

Descrição	Resultados Esperados
Testa abertura de uma área de trabalho informando-se um nome de arquivo inválido (arquivo já aberto, diretório inexistente ou acesso negado).	sistema exibe mensagem informando o erro
Testa abertura de uma área de trabalho informando-se nome de arquivo válido.	a área de trabalho é aberta

Caso de teste: Criar catalogador **Propósito:** Testar a corretude do Caso de Uso Criar catalogador	
Descrição	**Resultados Esperados**
Testa a inclusão de um novo processo de catalogação com dados inválidos (diretório de imagens não acessível, endereço do dicionário incorreto, catálogos não acessíveis)	sistema exibe mensagem de erro e não permite a inclusão do processo
Testa a inclusão de um novo processo de catalogação com dados válidos.	processo é incluído

Caso de teste: Editar catalogador **Propósito:** Testar a corretude do Caso de Uso Editar catalogador	
Descrição	**Resultados Esperados**
Testa a atualização de um processo de catalogação existente com dados inválidos (diretório de imagens não acessível, endereço do *gazetteer* incorreto, catálogos não acessíveis)	sistema exibe mensagem de erro e não permite a atualização do processo
Testa a atualização de um processo de catalogação existente com dados válidos.	processo é alterado

Caso de teste: Excluir catalogador

Propósito:	Testar a corretude do Caso de Uso Excluir catalogador
Descrição	**Resultados Esperados**
Testa a exclusão de um processo de catalogação existente	Processo é excluído

Caso de teste:	Iniciar catalogador
Propósito:	Testar a corretude do Caso de Uso Iniciar catalogador
Descrição	**Resultados Esperados**
Testa a interrupção automática de um processo de catalogação cuja configuração tornou-se inválida (diretório de imagens não acessível, endereço do dicionário incorreto, catálogos não acessíveis).	Processo deve ser interrompido e o seu status deve passar para “error”
Testa a catalogação de objetos.	novos objetos são catalogados

Caso de teste:	Parar catalogador
Propósito:	Testar a corretude do Caso de Uso Parar catalogador
Descrição	**Resultados Esperados**
Testa a interrupção do processo de catalogação	Processo deve ser interrompido e o seu status deve passar para “error”