4
Implementação do GeoCatalog

Esse capítulo apresenta o projeto detalhado do GeoCatalog, um protótipo de um software para catalogação automática de dados geográficos, como ilustração da arquitetura apresentada.

O projeto foi elaborado de forma a gerar como subproduto um *framework* para catalogação automática de dados genéricos (dados geográficos ou não) cujas características serão descritas na seção 4.6.

Assumimos as seguintes premissas de desenvolvimento:

1. Os dados geográficos são arquivos, contendo a descrição de uma imagem geográfica (um identificador e dois pares de coordenadas geográficas), em formato XML;
2. Os objetos relacionados à imagem são obtidos do ADL Gazetteer.
3. A interface de usuário é uma aplicação *desktop* desenvolvida em Java versão 1.5_06.
4. Não haverá comunicação remota entre os agentes. Todos os agentes envolvidos do processo de catalogação serão objetos em uma mesma aplicação.

A seção 4.1, com base nos requisitos funcionais identificados na seção 3.3, apresenta a descrição dos casos de uso do software. A seção 4.3 apresenta o projeto detalhado dos componentes da interface de usuário. A seção 4.4 apresenta a implementação da operação de inicialização de um processo de catalogação e a infra-estrutura para a implementação dos agentes. A seção 4.5 apresenta os componentes responsáveis por capturar descrições de um dado geográfico. E, por fim, a seção 4.6 generaliza o projeto para se adaptar a qualquer tipo de dado.

4.1. Casos de uso

O GeoCatalog utiliza o conceito de Área de Trabalho. Cada Área de Trabalho tem a função de repositório de configurações de processos de catalogação. Da mesma forma como softwares de edição de texto permitem abrir vários documentos em uma mesma instância de execução do editor, também é possível abrir várias Áreas de Trabalho para organizar convenientemente as configurações de catalogação.

Cada processo poderá ser alterado, iniciado e interrompido independentemente uns dos outros. Isso significa que, enquanto um processo pode estar ativo, outro pode não estar.

Nesse contexto, o diagrama de Casos de Uso da Figura 13 apresenta as funcionalidades disponíveis no GeoCatalog.

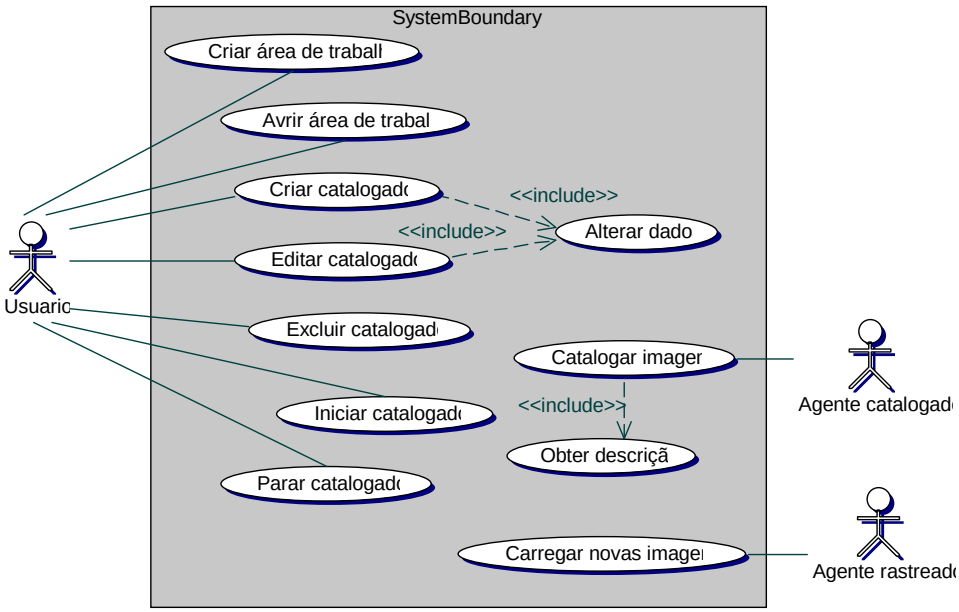

Figura 13 – Diagrama de Casos de Uso

A seguir, os quadros especificam cada um dos Casos de Uso identificados no diagrama.

Caso de Uso: Criar área de trabalho
Ator Primário: Usuário
Pré-condições: Nenhuma
Fluxo Normal 1 – Usuário seleciona local do arquivo 2 – Usuário Informa nome do arquivo 3 – Usuário confirma operação 4 – Sistema cria e abre a área de trabalho
Fluxo Alternativo não há

Caso de Uso: Abrir área de trabalho
Ator Primário: Usuário
Pré-condições: Nenhuma
Fluxo Normal 1 – Usuário seleciona o arquivo correspondente à área de trabalho desejada 2 – Usuário confirma seleção 3 – Sistema abre a área de trabalho selecionada
Fluxo Alternativo: Área de trabalho já está aberta 1a – Sistema informa que o arquivo já está aberto 1b – Sistema retorna ao passo 1 do fluxo normal

Caso de uso: Criar catalogador
Ator Primário: Usuário
Pré-condições: Área de trabalho aberta
Fluxo Normal 1 – Usuário escolhe solicita a inclusão de um novo processo 2 – Sistema executa **caso de uso Alterar Dados** 3 – Usuário solicita a salva da Área de Trabalho 4 – Sistema salva Área de Trabalho
Fluxo Alternativo: Existem locais onde os objetos serão procurados que não estão acessíveis 2a – Usuário corrige o nome dos locais não acessíveis 2b – Sistema retorna ao passo 3 do fluxo normal
Fluxo Alternativo: Existem enciclopédias não acessíveis 2a – Usuário corrige os dados de acesso às enciclopédias 2b – Sistema retorna ao passo 3 do fluxo normal
Fluxo Alternativo: O banco de dados não está acessível 2a – Usuário corrige os dados de acesso ao banco de dados do catálogo 2b – Sistema retorna ao passo 3 do fluxo normal

Caso de uso: Editar catalogador
Ator Primário: Usuário
Pré-condições: Área de trabalho aberta
Fluxo Normal 1 – Usuário solicita a alteração dos parâmetros de um processo de catalogação 2 – Usuário confirma alterações 3 – Sistema salva área de trabalho
Fluxo Alternativo: Existem locais onde os objetos serão procurados que não estão acessíveis 2a – Usuário corrige o nome dos locais não acessíveis 2b – Sistema retorna ao passo 3 do fluxo normal
Fluxo Alternativo: Existem enciclopédias não acessíveis 2a – Usuário corrige os dados de acesso às enciclopédias 2b – Sistema retorna ao passo 3 do fluxo normal
Fluxo Alternativo: O banco de dados não está acessível 2a – Usuário corrige os dados de acesso ao banco de dados do catálogo 2b – Sistema retorna ao passo 3 do fluxo normal

Caso de uso: Alterar dados
Ator Primário: Usuário
Pré-condições: Um dos casos de uso “Criar catalogador” ou “Editar catalogador” iniciados
Fluxo Normal 1 – Usuário altera parâmetros de configuração do catalogador (diretórios de imagens, URL ADL, nome do arquivo para o catálogo e nome do arquivo para as imagens com falhas) 2 – Usuário confirma informações
Fluxo Alternativo: não há

Caso de uso: Excluir catalogador
Ator Primário: Usuário
Pré-condições: Catalogador estar parado
Fluxo Normal 1 – Usuário seleciona catalogador 2 – Usuário escolhe opção “Remove process” 3 – Usuário confirma operação 4 – Sistema salva área de trabalho
Fluxo Alternativo: não há

Caso de uso: Iniciar catalogador
Ator Primário: Usuário
Pré-condições: Catalogador corretamente configurado
Fluxo Normal 1 – Usuário seleciona o catalogador desejado 2 – Usuário solicita a execução do catalogador
Fluxo Alternativo: Há algum erro de configuração no catalogador 1a – Usuário executa **caso de uso Editar Catalogador** 1b – Sistema retorna ao passo 1 do fluxo normal

Caso de uso: Parar catalogador
Ator Primário: Usuário
Pré-condições: Catalogador em execução
Fluxo Normal 1 – Usuário seleciona o catalogador desejado 2 – Usuário solicita interrupção do catalogador
Fluxo Alternativo: não há

Caso de uso: Carregar novas imagens
Ator Primário: Agente rastreador
Pré-condições: Nenhuma
Fluxo Normal 1 – Para cada objeto no local configurado no catalogador como fonte de imagens 1.1 – Agente obtém lista de arquivos disponíveis 1.2 – Para cada arquivo 1.2.1 – Se o arquivo não estiver catalogado nem estiver na lista de arquivos com falhas e nem no catálogo temporário 1.2.1.1 – Agente carrega metadados da imagem 1.2.1.2 – Agente adiciona objeto ao catálogo temporário 1.2.1.2 – Agente envia mensagem “Nova imagem” para o Agente Catalogador com os metadados
Fluxo Alternativo: Local fonte de objetos configurado no catalogador não está acessível 1.1a – Agente solicita ao processo de catalogação a interrupção de todos os agentes 1.1b – Sistema acusa erro de configuração

Caso de uso: Catalogar imagem
Ator Primário: Agente catalogador
Pré-condições: Nenhuma
Fluxo Normal 1 – Para cada mensagem “Nova imagem” 1.1 – Agente executa **caso de uso Carregar anotações** para a imagem enviada na mensagem 1.2 – Agente salva objeto no banco de dados do catálogo 1.3 – Agente retira imagem do catálogo temporário
Fluxo Alternativo: Não há anotações disponíveis para a imagem 1.1a – Agente salva objeto no banco de dados de falhas 1.1b – Agente continua o processamento com a próxima mensagem
Fluxo Alternativo: Não é possível salvar objeto no catálogo 1.2a – Agente solicita ao processo de catalogação a interrupção de todos os agentes 1.2b – Sistema acusa erro de configuração

Caso de uso: Obter descrição
Ator Primário: Agente catalogador
Pré-condições: Os *gazetteers* terem sido corretamente configurados
Fluxo Normal 1 – Recupera coordenadas da imagem 2 – Para cada *gazetteer* do catalogador 2.1 – Recupera *features* contidas na área geográfica coberta pela imagem 2.2 – Associa *features* à imagem
Fluxo Alternativo: *Gazetteer* configurado no catalogador não está acessível 2.1a – Agente solicita ao processo de catalogação a interrupção de todos os agentes 2.1b – Sistema acusa erro de configuração

4.2. Padrão arquitetural da aplicação

Arquitetura dos componentes do software

O projeto utilizou o padrão arquitetural Model View Controller (MVC) que propõe a divisão funcional dos componentes do software em três pacotes:

a. Modelo (*model*) → componentes encarregados de executarem as regras de negócio e controle de estado da aplicação.
b. Visão (*view*) → componentes para interface de usuário.
c. Controlador (*controller*) → componentes que mediam a comunicação entre componentes dos dois primeiros grupos. (segundo esse padrão não deve haver comunicação direta entre um componente do modelo e outro da visão)

Essa organização tem por finalidade aumentar a coesão dos componentes do software. Uma das suas principais vantagens é o aumento da flexibilidade nas manutenções, pois evita o entrelaçamento de procedimentos de regras de

negócios com de interface de usuário. Se, por exemplo, houver necessidade de substituir os mecanismos de interface de usuário, somente os componentes do pacote visão deverão sofrer manutenção. Se, por outro lado, for necessário alterar regras de negócio, então, somente as classes do pacote modelo serão alteradas. Os componentes de controle desempenham o papel de ligação entre os outros dois pacotes e só serão alterados quando houver modificação na interface de comunicação da visão com o modelo.

A Figura 14 mostra a divisão dos componentes do GeoCatalog segundo a arquitetura MVC. O pacote modelo foi dividido em duas partes para separar os componentes diretamente ligados com o processo de catalogação (pacote *model*) daqueles que controlam a aplicação, as áreas de trabalho e as configurações de processos (pacote *appmodel*).

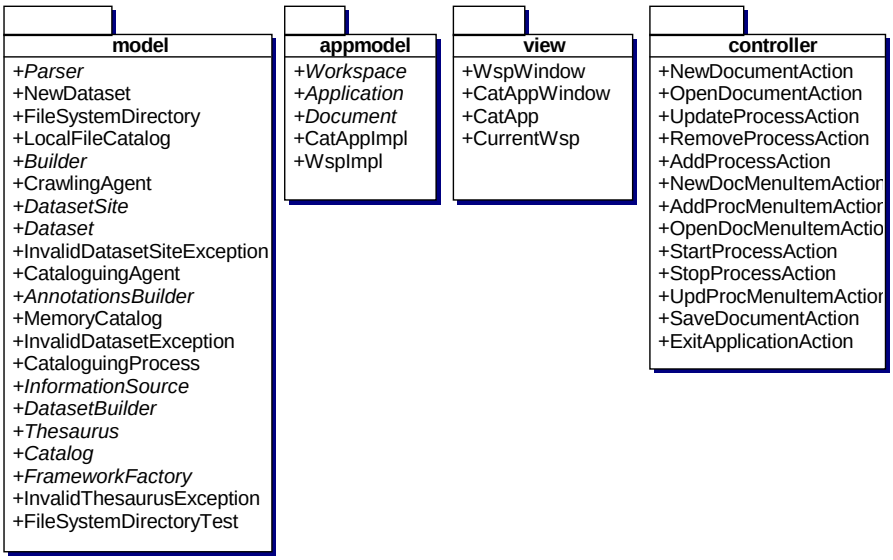

Figura 14 - Divisão de componentes do GeoCatalog segundo o padrão MVC

4.3. Interface de usuário

A Figura 15 apresenta os componentes de software da interface de usuário. Os componentes principais são as janelas principal (*CatAppWindow*) e de área de trabalho (*WspWindow*).

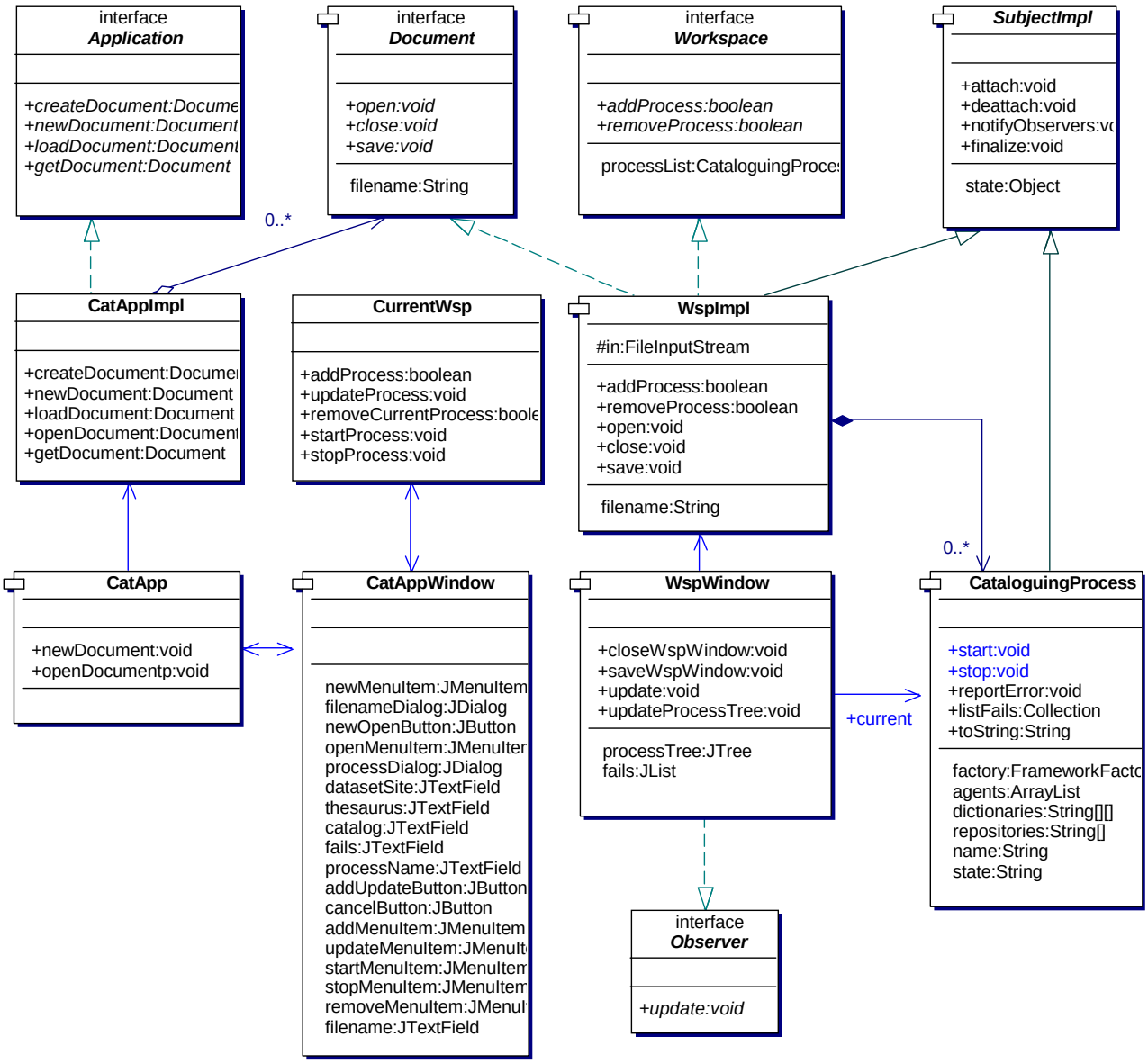

Figura 15 – Componentes da interface de usuário do GeoCatalog

A janela principal é a janela que inicia a aplicação e contém os componentes para manipulação das áreas de trabalho e processos de catalogação. São eles:

a. os itens de menu que executam os casos de uso do usuário;
b. uma janela de dialogo para obter o nome do arquivo de área de trabalho a ser criado ou aberto e
c. uma janela de diálogo para obter os parâmetros de configuração de um processo de catalogação: nome do processo, diretório para busca de imagens, endereço *Web* do ADL Gazetteer, nome do arquivo de catálogo e nome do arquivo de falhas.

A janela de área de trabalho (Figura 16) exibe informações sobre os processos de catalogação contidos na área de trabalho. Uma “árvore” de processos na metade à esquerda da “janela” exibe os nomes e o estado de cada processo de catalogação. Na metade à direita da “janela”, são listadas as falhas encontradas durante a execução do processo que estiver selecionado pelo usuário.

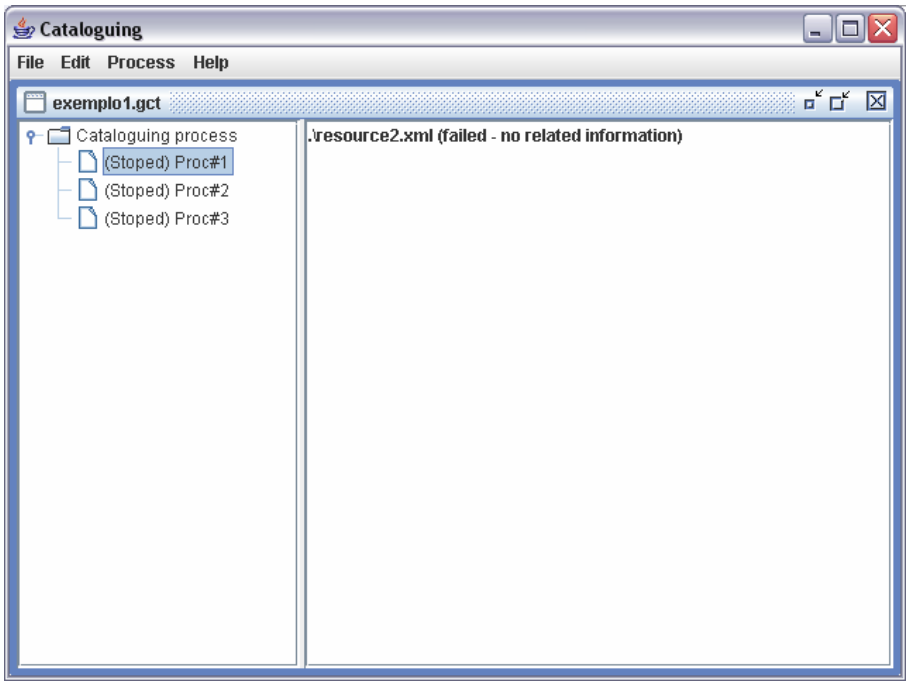

Figura 16 - Janela de área de trabalho

4.3.1. Interação com usuário

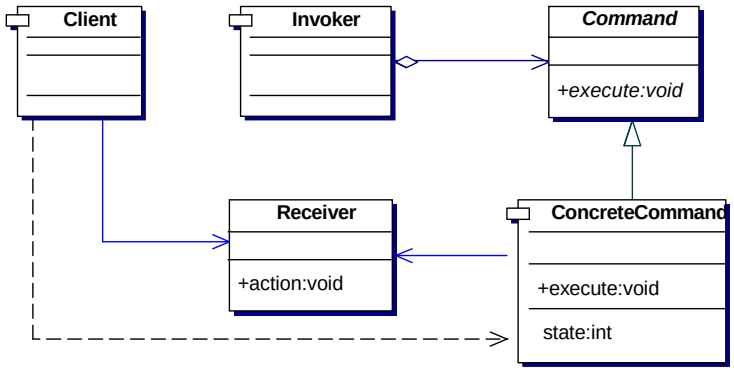

Figura 17 - Estrutura do padrão de projeto *Command*

Existem três interfaces de software: aplicação (*Application*), documento (*Document*) e área de trabalho (*Workspace*), que definem as operações disponíveis aos usuários. A primeira contém operações para a criação e carga de um documento[17]. A segunda contém operações sobre os documentos abertos. A última contém operações para manipulação de processos de catalogação em uma área de trabalho.

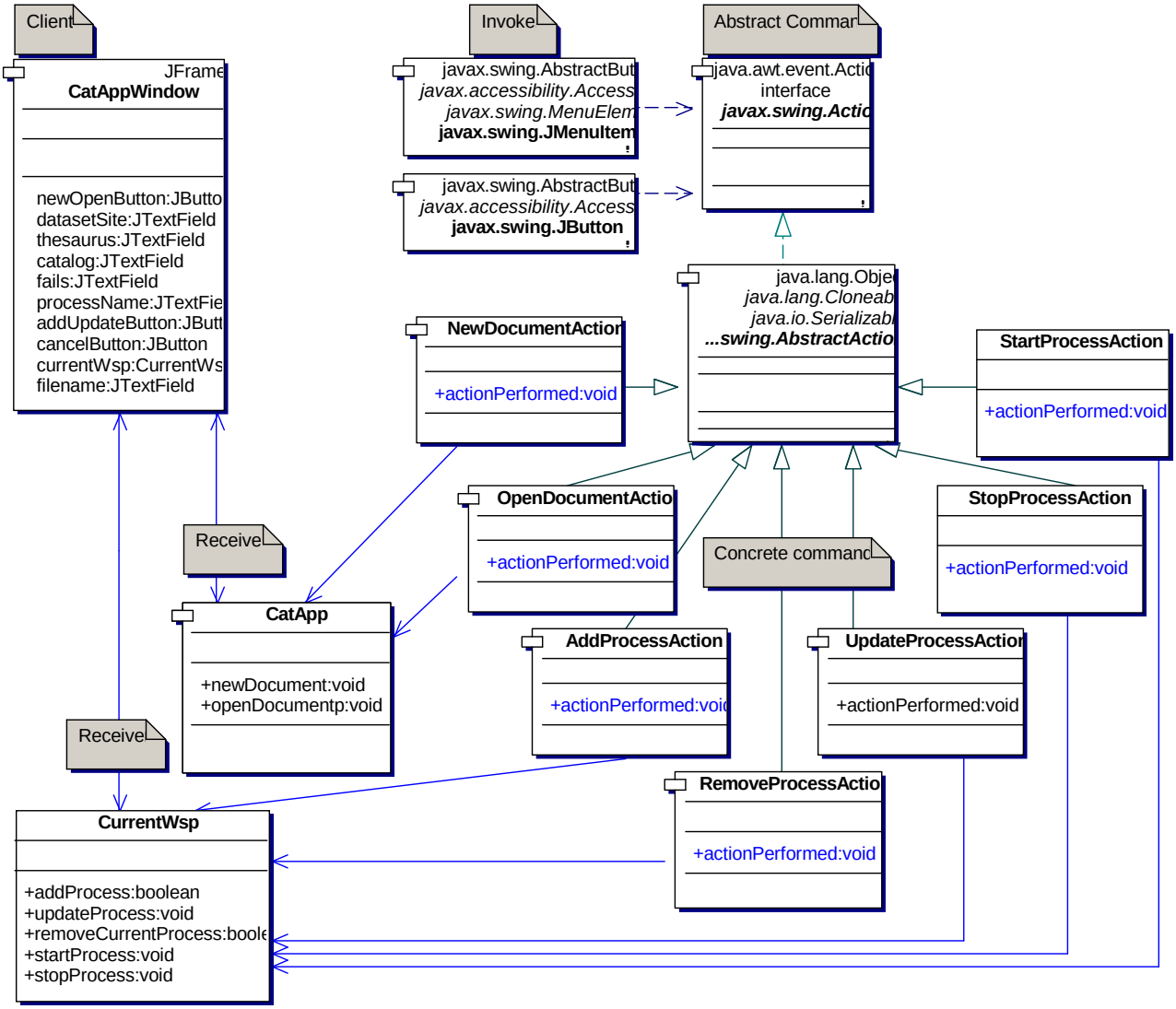

Figura 18 - Padrão *Command* no GeoCatalog

A implementação do acionamento das operações dessas interfaces foi baseada no padrão *Command*. Esse padrão tem como principal objetivo parametrizar a ação de um item de menu. A estrutura do padrão *Command*, segundo Erich Gamma [15], é apresentada na Figura 17.

[17] Documento para, essa aplicação, é uma área de trabalho.

Os componentes *CatAppImpl* e *WspImpl* do modelo implementam as operações das interfaces que são acionadas através dos receptores (*Receiver*) de comandos *CatApp* e *CurrentWsp* do pacote de controle, respectivamente. Como uma instância da aplicação pode conter várias áreas de trabalho abertas, o componente *CurrentWsp* executa as operações solicitadas, na área de trabalho que estiver selecionada pelo usuário. As operações dos receptores, por sua vez, são acionadas por comandos (*ConcreteCommand*) associados aos itens de menu da janela da aplicação.

A estrutura do padrão *Command* no GeoCatalog é apresentado na Figura 18.

4.3.2. Atualização das informações de interface de usuário

A janela de área de trabalho exibe uma lista com o nome e o estado[18] de todos os processos de catalogação nela contidos e uma lista das falhas de catalogação ocorridas no processo selecionado pelo usuário. Para manter tais informações atualizadas na janela foi utilizado o padrão Observer, cuja estrutura segundo Erich Gamma [15], é apresentada na Figura 19.

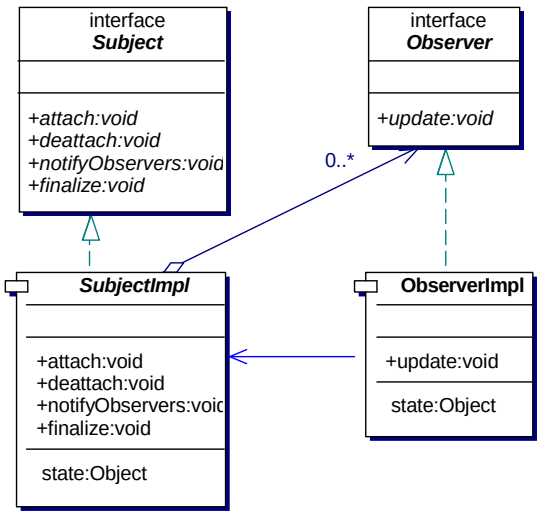

Figura 19 - Estrutura do padrão de projeto Observer

[18] Os estados possíveis de um processo de catalogação podem ser: executando, parado e interrompido por erro.

Nesse padrão objetos (*Subject*) são componentes cujas propriedades desejam ser monitoradas por um observador (*Observer*). Caso ocorra alguma alteração no conteúdo dessas propriedades, então, todos os observadores que se declararam interessados devem ser notificados. O funcionamento do mecanismo é o seguinte.

Em primeiro lugar, um observador se declara interessado no estado de um objeto solicitando a ele sua inclusão na lista de interessados. Para isso, o observador executa operação *attach(this)* do objeto passando como parâmetro uma referência para ele próprio. Quando o objeto sofre alguma alteração em seu estado, ele percorre a lista de observadores interessados e notifica cada um deles sobre a mudança. Essa notificação é implementada pela execução da operação *update( )* de cada observador na lista de interessados. Essa operação em cada observador irá solicitar do objeto o seu novo estado e executar os procedimentos adequados de atualização do observador. Nesse padrão toda a seqüência de troca de mensagens é síncrona.

Para compreendermos a vantagem desse padrão imaginemos outra estratégia para atualização de dados nos observadores. Se ao invés de receber notificações, o próprio observador, periodicamente, solicitasse aos objetos seu estado, o efeito prático seria o mesmo. Entretanto, poderia ocorrer que o observador solicitasse o estado do objeto muitas vezes sem que houvesse alteração alguma. Por outro lado, caso a freqüência com que um observador solicitasse informações sobre o estado fosse pequena em relação à freqüência com que ele se alterasse, muitos efeitos da mudança de estado seriam perdidos nos observadores.

No GeoCatalog a janela de Área de Trabalho é um observador dos processos de catalogação (*CataloguingProcess*), e da implementação de regras de negócio da área de trabalho (*WspImpl*). Quando há alteração do estado de um desses componentes o seu método *update( )* é executado para atualizar a lista de processos e de falhas do processo corrente. Os eventos que disparam a notificação da janela de área de trabalho são:

a. alteração do status do processo de catalogação para “*Running*” na sua inicialização

b. alteração do status do processo de catalogação para “*Stoped*” na sua interrupção
c. alteração do status do processo de catalogação para “Error” quando ocorre erro de comunicação ou de acesso a arquivo
d. alteração do nome do processo de catalogação
e. inclusão de nova falha de catalogação na lista
f. inclusão de processo de catalogação na Área de Trabalho
g. remoção de processo de catalogação da Área de Trabalho

4.3.3. Configuração de um processo de catalogação

Um processo de catalogação possui cinco parâmetros de configuração:

a. seu nome;
b. o nome de um diretório onde serão procurados novos arquivos de dados geográficos para serem catalogados;
c. o endereço de serviço de um dicionário geográfico;
d. o nome de um arquivo para catálogo;
e. o nome de um arquivo para catálogo de falhas.

Note que a limitação de um diretório de busca e um dicionário, tem a finalidade de simplificar a implementação. Essa limitação não está prevista na arquitetura apresentada anteriormente.

Tais informações serão armazenadas no processo de catalogação (*CataloguingProcess*) e mais tarde serão utilizadas para criar objetos que encapsulam funções de acesso ao conteúdo de arquivos, ao serviço de dicionário e ao conteúdo de catálogos, como veremos, mais adiante, na seção 4.4.

4.4. Inicialização de processo de catalogação

A inicialização de um processo de catalogação começa pela criação de seus agentes. Eles são recriados a cada vez que o processo é iniciado pela primeira vez em cada execução do GeoCatalog. O primeiro passo é criar os objetos que encapsulam as operações de acesso aos dados, dicionários e

catálogos (temporário, falhas e definitivo) que são *FileSystemDirectory*, *ADLGazetteer* e *LocalFileCatalog*, respectivamente.

Cada objeto é construído com uma fábrica de objetos que implementa o padrão Factory Method. Esse padrão é fundamental para permitir a customização do *framework* de catalogação que será discutido na seção 4.6.

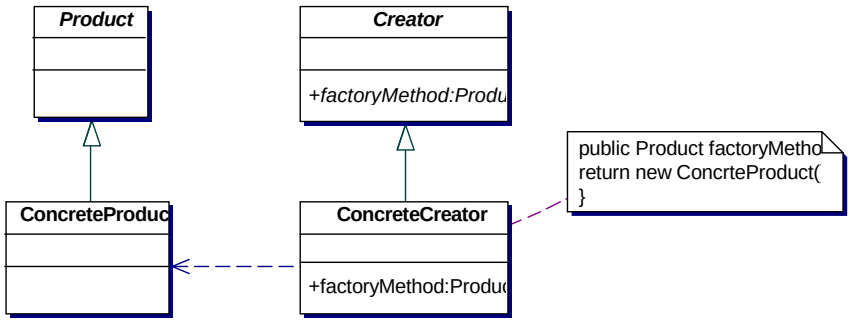

Figura 20 - Estrutura do padrão de projeto Factory Method

A estrutura básica do padrão Factory Method, segundo Erich Gamma [15], é apresentado na Figura 20. Interfaces abstratas da fábrica (*Creator*) e dos produtos (*Product*), conhecidas pelo *framework,* permitem utilizar, na implementação do comportamento do *framework*, todos os objetos que serão customizados pelo usuário. Quando os produtos concretos estiverem definidos, sua criação ficará delegada para uma fábrica concreta que implementa as operações abstratas de fábrica. No GeoCatalog a fábrica abstrata de objetos é *FrameworkFactory* (Figura 21) e a fábrica concerta é *CustomFactory*.

O próximo passo é a criação e inicialização dos agentes. Para criar o agente rastreador informa-se um identificador, uma lista de repositórios, um catálogo temporário, um catálogo de falhas e um catálogo definitivo. Para criar o agente catalogador informa-se os mesmos parâmetros exceto que a lista repositórios é substituída por uma lista de dicionários.

Agentes são componentes de software que executam *threads* diferentes e que podem se comunicar através de mensagens. Cada agente executa instruções específicas, mas os mecanismos para troca de mensagem e para execução das *threads* são genéricos e compartilhados entre todos. Por isso, foi definida uma infra-estrutura básica para comunicação e execução de cada agente.

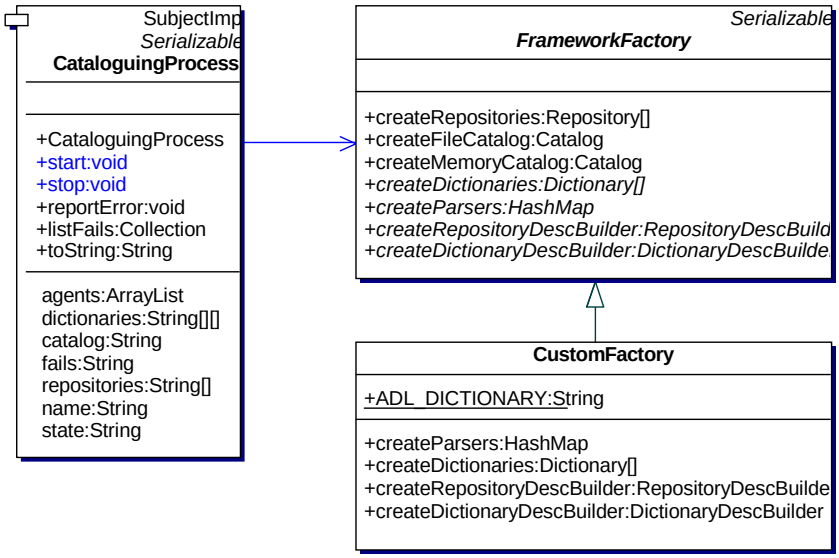

Figura 21 – Fábrica de objetos do GeoCatalog

A Figura 22 ilustra os componentes para a comunicação de agentes. O mecanismo baseia-se na troca de mensagens (*Message*) não endereçadas, ou seja, cada mensagem é enviada para todo o grupo de agentes através de um canal de comunicação (*MsgChannel*) comum. As mensagens específicas de cada aplicação serão subtipos da classe abstrata *Message*. O fluxo de processamento de cada agente é responsável por obter do canal de comunicação mensagens do tipo apropriado à sua atividade, interpretar apropriadamente o objeto genérico do seu corpo e dar tratamento a ele.

A implementação de um agente faz-se pela extensão da classe Agent que encapsula métodos de controle da execução (*start( )* e *stop( )*) e de comunicação (*sendMessage( )* e *readMessage(Message msg)*). O método abstrato *agentBehavior( )*, implementado no agente concreto, determinará o comportamento desse agente.

As funções de comunicação são delegadas à uma classe que implementa a interface *Comm*. Isso permite flexibilidade para, por exemplo, acrescentar comunicação distribuída ao sistema. Poder-se-ia criar uma classe *DistributedComm*, que implementasse *Comm*, para gerenciar a troca de mensagens entre componentes remotos. No GeoCatalog foi implementada somente a classe *LocalComm* para comunicação local entre os agentes.

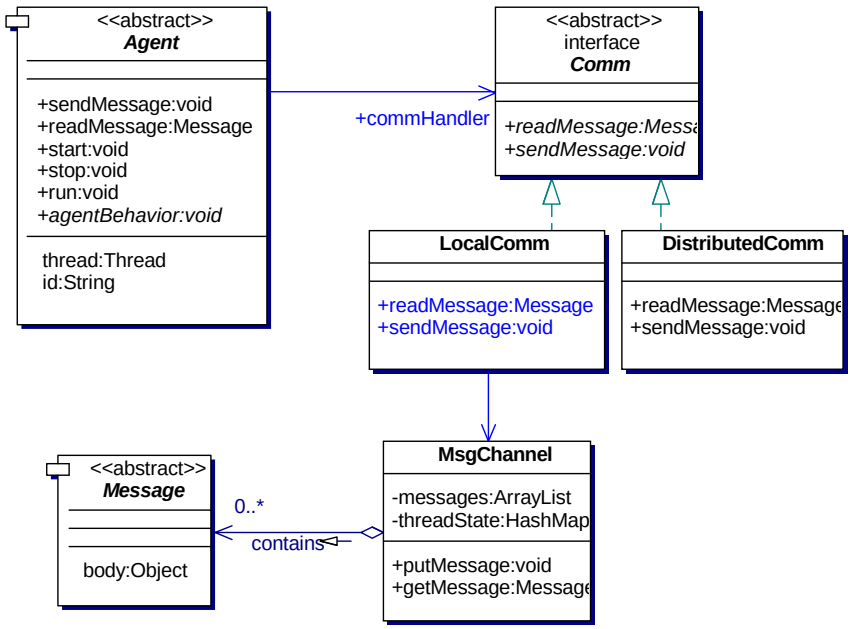

Figura 22 - Infra-estrutura para criação de agentes

No componente de comunicação local, implementado no protótipo, o canal de comunicação (*MsgChannel*) é protegido contra acessos simultâneos de dois agentes para evitar inconsistências do repositório de mensagens. Os métodos *sendMessage( )* e *readMessage( )* da interface de comunicação são sincronizados de modo que cada agente aguardará o canal estar desocupado para efetuar sua comunicação.

Como forma de otimizar processamento, existe um dispositivo que interrompe a *thread* de um agente caso, ao tentar recuperar uma mensagem, o canal de comunicação não retorne nenhuma mensagem. Sua atividade é reiniciada tão logo se acrescente uma nova mensagem no canal de comunicação.

4.5. Geração de metadados

No GeoCatalog os repositórios são implementados pela classe *FileSystemDirectory* que é responsável por extrair metadados de um arquivo armazenado no sistema de arquivos do computador. Sua operação *getDescription(String uri)* toma como parâmetro um nome de arquivo, decodifica seu conteúdo, extrai os metadados disponíveis, os armazena nas estruturas de metadados com o componente *OOISO19115GeoImage* e os devolve ao agente

rastreador. A identificação do tipo de decodificador apropriado ao dado é feita pela extensão do arquivo de dados. Os decodificadores são armazenados em uma coleção do repositório identificados pela extensão de arquivo que eles são capazes de decodificar.

Os arquivos contendo descrições de dados geográficos, de acordo com as premissas de projeto apresentadas no princípio do capítulo, são arquivos XML cujo esquema é o seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.inf.puc-
rio.br/geocatalog"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:gct="http://www.inf.puc-rio.br/geocatalog">
  <include schemaLocation="geometryBasic2d.xsd" />
  <element name="resource">
    <complexType>
      <sequence>
        <element name="identifier" type="string"
        minOccurs="1" maxOccurs="1">
        </element>
        <element name="bounding-box"
        type="gml:EnvelopeType"
          minOccurs="1" maxOccurs="1">
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```

Os metadados extraídos do repositório são:

1. URI do repositório da imagem;
2. identificador da imagem;

3. coordenadas geográficas dos pontos da diagonal do retângulo de contorno da imagem.

Os dicionários são implementados pela classe *ADLGazeteer* que é responsável por extrair objetos do dicionário geográfico ADL Gazetter. Sua operação *getDescription(DatasetDescription description)* toma como parâmetro a descrição de dado obtida do repositório, elabora uma consulta ao dicionário, conforme descrito na seção 3.1, armazena os metadados obtidos, também, com o componente *OOISO19115GeoImage* e devolve o resultado para o agente catalogador.

A consulta ao dicionário ADL é codificada segundo o protocolo de serviço ADL[19]. Segundo a estratégia de catalogação apresentada serão recuperados os objetos do dicionário contidos na área geográfica retangular definida pelas coordenadas da imagem, cujos valores foram armazenados na descrição do repositório.

A consulta ao dicionário assume a seguinte forma:

[19] Vide Apêndice A – Protocolo de serviço do ADL Gazetteer (p. 101).

```
<?xml version="1.0" encoding="UTF-8"?>
<gazetteer-service
xmlns=http://www.alexandria.ucsb.edu/gazetteer
xmlns:gml="http://www.opengis.net/gml" version="1.2">
  <query-request>
    <gazetteer-query>
      <footprint-query operator="within">
        <gml:Box>
          <gml:coordinates>
           Longitude esquerda superior, Latitude esquerda
      superior, Longitude direita inferior, Latitude
      direita inferior
          </gml:coordinates>
        </gml:Box>
      </footprint-query>
    </gazetteer-query>
    <report-format>standard</report-format>
  </query-request>
</gazetteer-service>
```

Os metadados extraídos do dicionário são (para cada objeto):

1. Identificador do objeto no dicionário;
2. Nome de exibição da objeto;
3. coordenadas geográficas do objeto.

4.6. Framework de catalogação de dados

Um *framework* é um projeto reutilizável de software. Define um conjunto de classes e um padrão de colaboração entre elas para um domínio específico de aplicação. Por exemplo, um *framework* para editores gráficos pode ser criado independentemente do tipo de editor que se deseja implementar. Editores de partituras musicais, circuitos eletrônicos e desenhos, são exemplos de aplicações que poderiam utilizá-lo.

Para permitir tal generalidade deve-se determinar quais pontos do projeto podem ser customizados em cada tipo de aplicação e para cada um deles definir

interfaces abstratas. O projeto do *framework* utiliza tais interfaces genéricas para compor a colaboração entre as classes e, assim, definir o comportamento básico dos componentes. Ao aplicá-lo na construção de um software específico, as operações, antes abstratos, irão incorporar o comportamento adequado para o tipo de aplicação.

De acordo com Pree (1994) [16], um *framework* consiste em *frozen spots* e *hot spots*. *Frozen spots* são as classes e colaborações pré-definidas que não podem ser alteradas na customização. *Hot spots* são os pontos de extensão, pontos abstratos no projeto que serão customizados. Analogamente a uma instância de uma classe diz-se que um *framework* customizado para uma determinada aplicação é uma instância do *framework* original.

O princípio fundamental que norteia o projeto desses artefatos de software é o seguinte: “Não nos chame. Chamaremos você” (Larman 2002). Isso significa que as classes definidas pelo usuário não devem chamar as classes pré-definidas. Elas receberão chamadas destas.

Com a abordagem descrita é possível aumentar o grau de confiabilidade do software e diminuir o esforço de desenvolvimento, uma vez que código fonte e até mesmo código objeto, já testados, são reutilizados.

Framework para catalogação de dados genéricos

Da mesma forma como um grande volume de dados geográficos está disponível em grande variedade de repositórios e formatos, dados de outras naturezas, como áudio e vídeo, imagens e documentos também estão e compartilham os mesmos problemas de localização. A recuperação de alguns desses dados, no entanto, apresenta problemas adicionais. Dados multimídia, por exemplo, necessitam de sincronismo e velocidade constante de recuperação para a sua correta apresentação. Nesse trabalho, não trataremos dessa última categoria de problemas. Concentraremos a atenção no problema comum de localização do dado.

A solução do problema para todos os tipos de dados também são catálogos de metadados. Um exemplo da utilização desses catálogos em outro domínio de aplicação é apresentado por Hollink et al. [4] em *Semantic*

Annotation of Image Collections, onde eles descrevem uma técnica para descrição de uma coleção de arte. Surge, então, a idéia de um *framework* para catalogação automática de dados, onde o fluxo de processamento: obter novos dados, obter metadados e catalogar dado, é comum em todas as áreas aplicação. As regras de geração dos metadados, no entanto, devem ser customizadas em cada caso. É claro que só faz sentido falarmos em *framework* de catalogação automática se houver algum algoritmo para geração automática de metadados. No caso dos dados geográficos, vimos que dicionários geográficos capazes de identificar objetos contidos em determinada área geográficas fornece um mecanismo eficiente para a geração automática de metadados. Teoria análoga deve ser apresentada para cada domínio de aplicação do *framework*. Nesse trabalho, no entanto, não abordaremos essa questão.

A Figura 23 apresenta um esquema do *framework* para catalogação automática de dados.

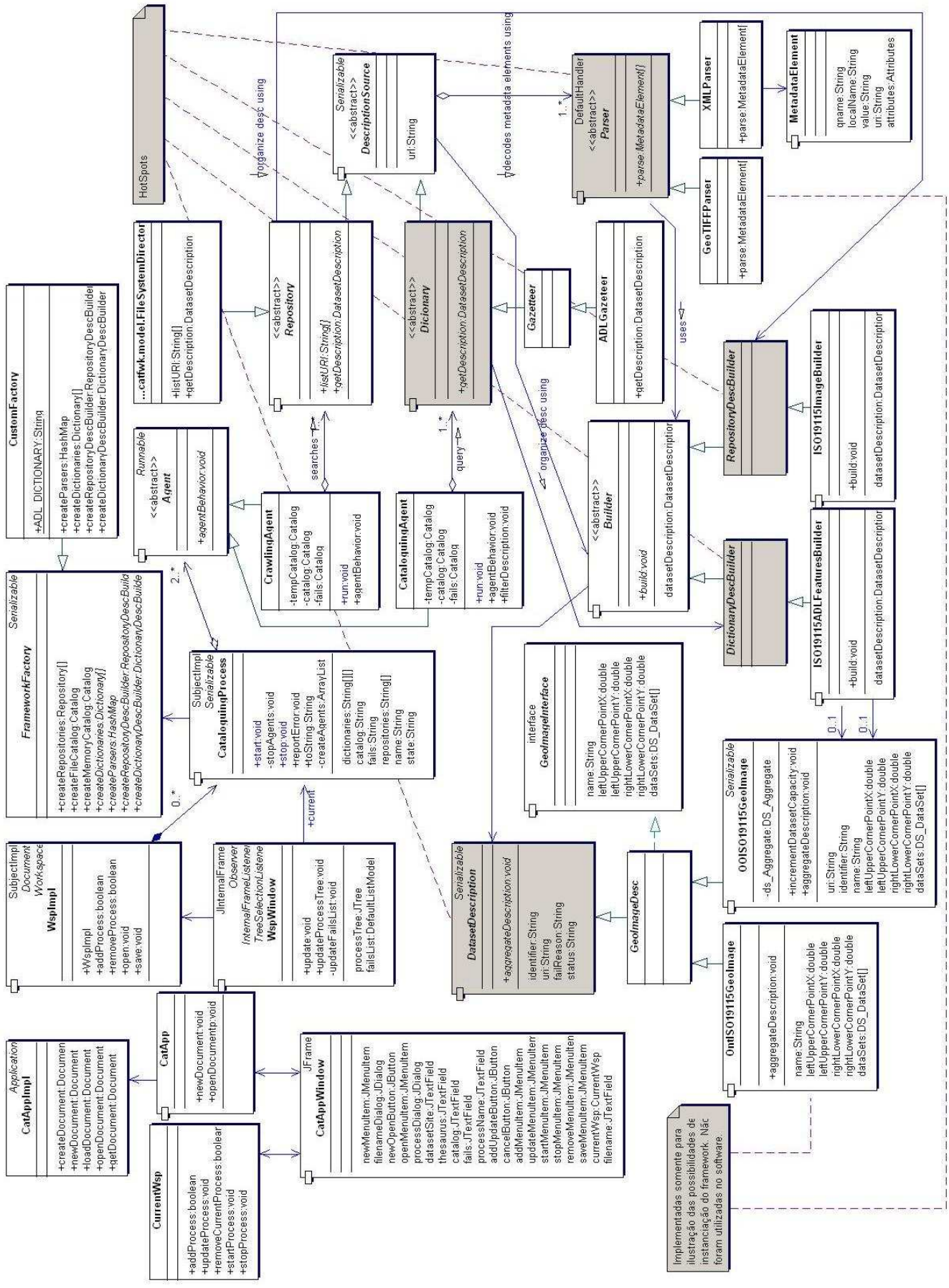

Figura 23 – *Framework* de catalogação automática

Os *frozens spots* desse *framework* são determinados pelo processo de catalogação e interface de usuário que podem ser reutilizados nas diferentes áreas de aplicação.

Os *hot spots*, que devem ser customizados pelos usuários são os seguintes:

a. Estrutura para armazenamento de metadados;

A classe *DatasetDescription* é a classe genérica que representa os metadados do dado a ser catalogado. Define propriedades básicas como identificador e URI (identificador do recipiente do dado) e um único método para agregação de objetos relacionados passados como parâmetro.

```
public abstract class DatasetDescriptin {
    private String identifier = null;
    private String uri = null;
    public abstract void aggregateDescription
    (DatasetDescription datasetDescription);
    public String getIdentifier( ) {
        return identifier;
    }
    public void setIdentifier(String identifier) {
        this.identifier = identifier;
    }
    public String getUri( ) {
        return uri;
    }
    public void setUri(String uri) {
        this.uri = uri;
    }
}
```

No GeoCatalog essa classe foi estendida pela classe *GeoImage* que implementa outra interface para capturar metadados de arquivos de

descrição de imagens geográficas. Subclasses de *GeoImage* determinam a forma de armazenamento dos metadados. Em *OOISO19115GeoImage* os metadados são armazenados em estruturas orientadas a objeto. Ela é a extensão de *DatasetDescription* realmente utilizada na aplicação. Em *OntISO19115GeoImage* os metadados são armazenados segundo a norma ISO19115, porém, utilizando-se um modelo semântico definido em uma ontologia. Ela foi implementada somente a título de ilustração das variações possíveis na instanciação desse *framework*.

b. Os dicionários de termos;

A classe *Dictionary* é a classe genérica que representa o ponto de acesso aos dados de um dicionário. Define um único comportamento para obtenção de metadados que recebe como parâmetro metadados obtidos de um repositório e devolve objetos relacionados a eles armazenados em um dicionário.

```
public abstract class Dictionary extends
DescriptionSource{
    Private transient DictionaryDescBuilder builder =
    null;
    public Dictionary(String url) {
        super(url);
    }
    public abstract Dataset getDescription
    (DatasetDescription datasetDescriptin);
    public Builder getBuilder( ) {
        if (builder == null){
            builder = getFactory( ).
            createDictionaryDescBuilder( );
            return builder;
        }
    }
}
```

No GeoCatalog essa classe foi estendida em *ADLGazetteerService* que implementa a consulta ao dicionário geográfico ADL.

c. A codificação com que os metadados e as consultas aos dicionários são fornecidos ao software.

Os procedimentos de geração de metadados baseia-se, conforme apresentado na seção 3.4, em dois componentes: *Parser Builder*. O primeiro, é utilizado na decodificação do conteúdo de um repositório ou das informações obtidas de um dicionário. Define uma única operação para obtenção das informações: *parse( ).*

```
public abstract class Parser extends DefaultHandler {
    private Builder builder = null;
    public Parser(Builder builder) {
        this.builder = builder;
    }
    public abstract MetadataElement[ ] parse (InputSource
    source);
    public Builder getBuilder( ) {
        return builder;
    }
}
```

Na instância para catalogação de dados geográficos assumimos que tanto os dados geográficos quanto o protocolo de serviço do dicionário forneceriam informações em XML. Por isso, foi implementado somente um *XMLParser*. Como ilustração de variações possíveis, foi criado o componente *GeoTIFFParser* que seria responsável por encapsular algoritmos de decodificação de arquivos de imagens geográficas em formato GeoTIFF.

O segundo componente, o *Builder*, tem dois propósitos: construir um descrição de dados para armazenar os metadados extraídos do repositório do dado (*RepositoryDescBuilder*) e construir uma descrição de dados para armazenar objetos obtidos de um dicionário (*DictionaryDescDescription*).

```
public abstract class Builder {
    public abstract void build(Object
    datasetElements);
    public abstract Dataset getDataset( );
}
```

No GeoCatalog *RepositoryDescBuilder* foi estendida em *ISO19115ImageBuilder* que constrói um GeoImage a partir de um vetor *MetadataElement*[ ], segundo o esquema de metadados definido pela ISO19115. A DictionaryDescBuilder foi estendida em *ISO19115ADLFeaturesBuilder* que constrói, também a partir de um vetor com a mesma estrutura, um *GeoImage* contendo objetos obtidas de um dicionário, segundo esquema da ISO19115.

Para que o *framework* seja capaz de instanciar as classes definidas pelo usuário, uma fábrica de objetos, FrameworkFactory, baseada no padrão de projeto Factory Method foi utilizada. Os métodos abstratos foram implementados na classe de usuário CustomFactory.

4.7. Métricas e organização do código

A seguir a listagem mostra os pacotes de organização das classes Java e algumas métricas de código do GeoCatalog.

Legenda:
LOC = linhas de código
NOC = número de classes
NOA = número de atributos
NOO = número de operações

Item	LOC	NOA	NOC	NOO
<total>	4170	47	137	45
br	3820	47	81	45
br.pucrio	3820	47	81	45
br.pucrio.inf	3820	47	81	45

br.pucrio.inf.catfwk	2216	47	43	45
br.pucrio.inf.catfwk.appmodel	132	3	5	8
br.pucrio.inf.catfwk.controller	241	1	13	3
br.pucrio.inf.catfwk.model	889	10	21	22
br.pucrio.inf.catfwk.view	954	47	4	45
br.pucrio.inf.geocat	1271	5	17	22
br.pucrio.inf.lib	309	4	20	10
br.pucrio.inf.lib.agent	123	4	6	10
br.pucrio.inf.lib.command	56	2	5	5
br.pucrio.inf.lib.factoryMethod	16	0	4	1
br.pucrio.inf.lib.observer	61	2	4	6
br.pucrio.inf.lib.util	53	0	1	4
iso	350	27	56	1
iso.iso19109	15	3	4	0
iso.iso19109.metadata	15	3	4	0
iso.iso19115	253	27	40	0
iso.iso19115.appliation_schema				
iso.iso19115.application_schema	3	0	1	0
iso.iso19115.citation	82	13	12	0
iso.iso19115.constraint	3	0	1	0
iso.iso19115.content	3	0	1	0
iso.iso19115.data_quality	3	0	1	0
iso.iso19115.distribution	3	0	1	0
iso.iso19115.identification	48	8	8	0
iso.iso19115.maintenance	6	0	2	0
iso.iso19115.metadata	43	27	1	0
iso.iso19115.metadata_extension	7	1	2	0
iso.iso19115.portrayal_catalogue	3	0	1	0
iso.iso19115.reference				
iso.iso19115.reference_system	9	2	2	0
iso.iso19115.spatial_representation	40	7	7	0
iso.iso19139	69	3	9	1
iso.iso19139.dataset	69	3	9	1

4.8. Exemplo de utilização

Como exemplo de catalogação, tomemos um repositório de dados geográficos que contém três arquivos arquivos XML como definidos a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<resource xmlns:gml="http://www.opengis.net/gml">
     <identifier>Rio de Janeiro (região do Maracanã)
</identifier>
     <bounding-box>
          <gml:coord>
               <gml:X>-43.26</gml:X>
               <gml:Y>-22.891</gml:Y>
          </gml:coord>
          <gml:coord>
               <gml:X>-43.203</gml:X>
               <gml:Y>-22.932</gml:Y>
          </gml:coord>
     </bounding-box>
</resource>
```

Figura 24 – Arquivo contendo a descrição de área geográfica do entorno do estádio do Maracanã (Rio de Janeiro)

```
<?xml version="1.0" encoding="UTF-8"?>
<resource xmlns:gml="http://www.opengis.net/gml">
      <identifier>Região vazia (coordenadas repetiodas)
</identifier>
      <bounding-box>
            <gml:coord>
                  <gml:X>-42.8</gml:X>
                  <gml:Y>-22.2</gml:Y>
            </gml:coord>
            <gml:coord>
                  <gml:X>-42.8</gml:X>
                  <gml:Y>-22.2</gml:Y>
            </gml:coord>
      </bounding-box>
</resource>
```

Figura 25 – Arquivo contendo a descrição de uma região vazia

```
<?xml version="1.0" encoding="UTF-8"?>
<resource xmlns:gml="http://www.opengis.net/gml">
      <bounding-box>
            <gml:coord>
                  <gml:X>-42.8</gml:X>
                  <gml:Y>-22.2</gml:Y>
            </gml:coord>
            <gml:coord>
                  <gml:X>-42.8</gml:X>
                  <gml:Y>-22.2</gml:Y>
            </gml:coord>
      </bounding-box>
</resource>
```

Figura 26 – Arquivo com erro de estrtura (falata o identificador)

Definamos um processo de catalogação com a seguinte configuração:

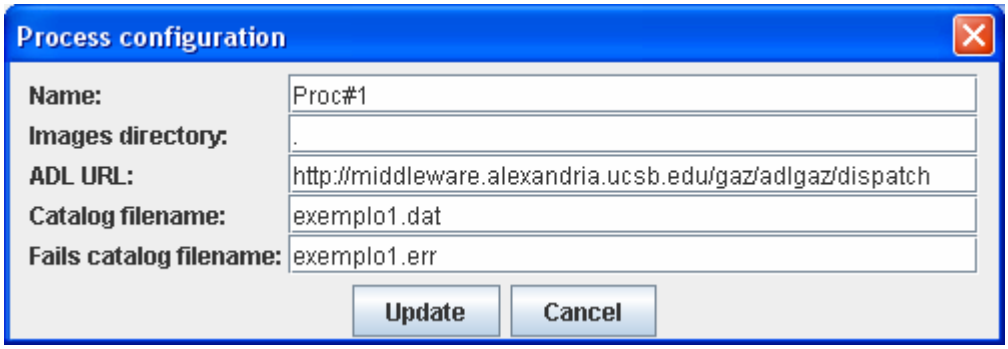

Figura 27 – Configuração de processo de catalogação no GeoCatalog

Os parâmetros de configuração do processo de catalogação acima definem que os dados geográficos serão procurados no diretório corrente, e o gazetteer a ser utilizado é o ADL Gazetteer. Além disso, são informados os nomes dos arquivos de catálogos para os dados que sofrerem alguma falha no processo de geração de metadados e para aqueles sem nenhuma exceção.

Ao executarmos esse processo sobre os três arquivos anteriores será produzida uma imagem do catálogo de dados geográficos (exemplo1.dat), em formato XML, contendo somente os metadados do primeiro arquivo XML e os objetos do gazetteer contidos na região definida pelo arquivo. Os demais arquivos não poderão ser catalogados uma vez que apresentam irregularidades. O segundo arquivo contém a descrição de uma região espacial vazia, ou seja, correspondente a um ponto. Por isso, não será possível encontrar, no Gazetteer, nenhum objeto contido nessa região, a não ser que seu ponto de representação coincida com aquele descrito no arquivo, o que não é o caso nesse exemplo. O terceiro arquivo contém uma falha estrutural, não possui o identificador (tag <\identifier> e será desprezado sem que o processo tente obter alguma informação do ADL Gazetteer.

Assim, o arquivo produzido pelo processo de catalogação será como mostrado a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
  <resource xmlns:gml="http://www.opengis.net/gml">
    null
    <identifier>Maracana</identifier>
    <bounding-box>
      <gml:coord>
        <gml:X>-43.26</gml:X>
        <gml:Y>-22.891</gml:Y>
      </gml:coord>
      <gml:coord>
        <gml:X>-43.24</gml:X>
        <gml:Y>-22.932</gml:Y>
      </gml:coord>
    </bounding-box>
    <features>
      <feature>
        null
        <identifier>adlgaz-1-1428551-52</identifier>
        <display-name>Engenho Novo, Serra do – Brazil
</display-name>
        <bounding-box>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
        </bounding-box>
      </feature>
      <feature>
        null
        <identifier>adlgaz-1-1456642-6b</identifier>
        <display-name>Riachuelo, Estacao – Brazil
</display-name>
        <bounding-box>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
        </bounding-box>
      </feature>
```

Figura 28 – Metadados produzidos pela catalogação dos arquivos de dados anteriormente especificados

```
<feature>
        null
        <identifier>adlgaz-1-1457206-44</identifier>
        <display-name>Rocha, Estacao – Brazil </display-
name>
        <bounding-box>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
        </bounding-box>
      </feature>
      <feature>
        null
        <identifier>adlgaz-1-1461061-73</identifier>
        <display-name>
          Sao Francisco Xavier, Estacao - Brazil
        </display-name>
        <bounding-box>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
        </bounding-box>
      </feature>
      <feature>
        null
        <identifier>adlgaz-1-1434239-62</identifier>
        <display-name>Heredia de Sa - Brazil</display-
name>
        <bounding-box>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
          <gml:coord>
            <gml:X>-43.25</gml:X>
            <gml:Y>-22.9</gml:Y>
          </gml:coord>
        </bounding-box>
      </feature>
    </features>
  </resource>
</resources>
```

Figura 29 – Metadados produzidos pela catalogação dos arquivos de dados anteriormente especificados (cont.)