



**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE CAMPOS**  
Universidade da Tecnologia e do Trabalho



## **CURSO SUPERIOR DE TECNOLOGIA EM DESENVOLVIMENTO DE SOFTWARE**

**RONALDO AMARAL SANTOS**

### **GRANULATEO3TOOL UMA FERRAMENTA PARA GRANULARIZAÇÃO DE DOCUMENTOS**

Campos dos Goytacazes/RJ

2007



RONALDO AMARAL SANTOS

GRANULATEO3TOOL  
UMA FERRAMENTA PARA GRANULARIZAÇÃO DE DOCUMENTOS

Monografia apresentada ao Centro Federal de Educação Tecnológica de Campos como requisito parcial para conclusão do Curso Superior de Tecnologia em Desenvolvimento de Software.

Orientador: Prof. Dr. Rogério Atem de Carvalho

Campos dos Goytacazes/RJ

2007

RONALDO AMARAL SANTOS

GRANULATEO3TOOL  
UMA FERRAMENTA PARA GRANULARIZAÇÃO DE DOCUMENTOS

Monografia apresentada ao Centro Federal de Educação Tecnológica de Campos como requisito parcial para conclusão do Curso Superior de Tecnologia em Desenvolvimento de Software.

Aprovada em 24 de setembro de 2007

Banca Avaliadora:

---

Profº Rogério Atem de Carvalho (orientador)  
Doutorado em Engenharia de Produção – UENF  
Centro Federal de Educação Tecnológica/Campos/RJ

---

Profº Jefferson Manhães de Azevedo  
Mestrado em Informática – UFRJ  
Centro Federal de Educação Tecnológica/Campos/RJ

---

Profº William da Silva Vianna  
Doutorado em Engenharia e Ciência dos Materiais – UENF  
Centro Federal de Educação Tecnológica/Campos/RJ

## **AGRADECIMENTOS**

Agradeço aos meus pais Donaldo Martins dos Santos e Maria de Fátima Amaral Santos pelo esforço de garantir minha formação e sem esquecer também da minha irmã Ronia

Agradeço ao CEFET Campos por me proporcionar um espaço de ensino de qualidade e em especial ao Núcleo de Pesquisa em Sistemas de Informação - NSI.

Não podia deixar de agradecer também a galera do NSI, pelos momentos de compreensão durante a elaboração deste trabalho e pela enorme contribuição que tiveram no desenvolvimento do GranulateO3Tool, sem o conhecimento adquirido com esses colegas este trabalho seria inviabilizado.

Agradeço ao professor e nosso coordenador Rogério Atem de Carvalho, pela orientação prestada neste trabalho e por ter me condicionado a oportunidade de participar do NSI.

“Prática é quando tudo funciona mas a gente não sabe porque.

Teoria é quando nada funciona mas a gente sabe porque.

Neste recinto se conjugam teoria e prática:

Nada funciona e ninguém sabe porquê...”

*Anônimo*

## RESUMO

A ferramenta desenvolvido neste trabalho faz parte de uma solução em *Enterprise Content Management* (ECM) totalmente baseada em software livre e que inclui todo o ferramental de Gestão de Conteúdo, *Groupware*, *Personal Information Management* (PIM), Granularização de Documentos e Objetos de Aprendizagem, desenvolvida no Núcleo de Pesquisa em Sistemas de Informação – NSI, CEFET Campos. A ferramenta conceitua-se em reusabilidade total ou parcial de conteúdo, sendo este um dos principais requisitos dos sistemas baseados em conteúdo. Tal requisito vem de encontro ao conceito de objetos de aprendizagem, onde este é um “grão” de conteúdo autônomo que, associado a metadados, poderá ser reutilizado em contextos e situações diferentes. A granularização permite desdobrar o conteúdo relativo a um assunto em vários tópicos que podem ser recombinaados em diferentes recursos. Diante deste cenário desenvolveu-se o GranulateO3Tool, uma ferramenta capaz de tratar informação semi-estruturada, identificando grãos ou fragmentos de texto através do processo de granularização. Esta ferramenta foi aplicada num estudo de caso envolvendo requisitos de um ambiente para Ensino à Distância, onde se pode concluir o caráter inovador da ferramenta e seus benefícios como um instrumento de apoio a geração e compartilhamento do conhecimento.

Palavras-chave: Sistemas de Gestão de Conteúdo. Enterprise Content Management. Granularização de Documentos. Objetos de Aprendizagem.

## **ABSTRACT**

The goal of this work is to present an Enterprise Content Management (ECM) solution based in free software that is comprised of a complete toolset for Content Management, Groupware, Personal Information Management (PIM), Document Granulation, and Learning Objects; developed by the Information Systems Research Group - ISrg, CEFET Campos. The tool is conceptually based on total or partial content reuse, being this one the main requirement for content based systems. Such requirement goes towards the learning object concept - a “grain” of autonomous content that, associated to methods, can be reused in different contexts and situations. Granulation helps to unfold the content - relative to a certain subject - that can be recombined to form new information resources. Leading this process is the GranulateO3Tool, capable of treating semi-structured information by identifying grains or fragments of text. This tool was applied in a case study involving a Distance Education environment, making one can identify the innovative character of the tool and its benefits as a support instrument for knowledge generation.

Keywords: Content Management. Enterprise Content Management. Content Granulation. Learning Objects.



## LISTA DE FIGURAS

Figura 1: Pirâmide da agregação do valor à informação.....	4
Figura 2: Interface de execução de scripts Python OpenOffice.org.....	15
Figura 3: Scripts Python utilizando o PyUno - Fonte: OPENOFFICE.ORG.....	16
Figura 4: Scripts Python como Componentes do UNO - Fonte: OPENOFFICE.ORG.....	16
Figura 5: Esquema de comunicação servidor ZOPE e ood.....	17
Figura 6: Exemplo de arquivo XML.....	18
Figura 7: Interface de configuração GranulateO3Tool.....	20
Figura 8: Deployment Diagram GranulateO3Tool.....	21
Figura 9: Diagrama de Classe GranulateO3Tool.....	22
Figura 10: Trecho de código função de conversão de arquivos.....	23
Figura 11: Método de retirada de imagem getImageDocumentList.....	24
Figura 12: Trecho de código XML representando os estilos do documento.....	26
Figura 13: Trecho de código XML representando o elemento Tabela.....	27
Figura 14: Métodos para retirada dos estilos de um elemento Tabela.....	27
Figura 15: Algoritmo de geração automática de documentos - docBuilderO3.....	30
Figura 16: Edição via Web de conteúdos utilizando-se o editor Kupu.....	31
Figura 17: action resultado de busca - addGrainCollectorSearch.....	32
Figura 18: Script Python addGrainCollector.....	32
Figura 19: Diagrama de Classe do tipo de conteúdo Document.....	36
Figura 20: Processo de Granularização de Documentos.....	38
Figura 21: Tela de Inserção de Arquivo.....	39
Figura 22: Tela de Edição de Metadados.....	40
Figura 23: Tela de exibição de documentos.....	41
Figura 24: Tela de visualização do grão Tabela.....	42
Figura 25: Tela de visualização do grão Imagem.....	42

Figura 26: Portlet GrainCart.....	43
Figura 27: Tela OpenOffice.org, tabela gerada a partir do docBuilderO3.....	44

## LISTA DE ABREVIATURAS E SIGLAS

<b>ABNT</b>	- Associação Brasileira de Normas Técnicas
<b>API</b>	- <i>Application Programming Interface</i>
<b>ASP</b>	- <i>Active Server Pages</i>
<b>CMF</b>	- <i>Content Management Framework</i>
<b>CMS</b>	- <i>Content Management Systems</i>
<b>DOM</b>	- <i>Document Object Model</i>
<b>DTML</b>	- <i>Document Template Markup Language</i>
<b>EaD</b>	- Ensino à Distância
<b>ECM</b>	- Enterprise Content Management
<b>GNU</b>	- <i>General Public License</i>
<b>GPL</b>	- <i>General Public License</i>
<b>HTML</b>	- <i>HyperText Markup Language</i>
<b>JSP</b>	- <i>JavaServer Pages</i>
<b>LMS</b>	- <i>Learning Management System</i>
<b>LOM</b>	- <i>Learning Object Metadata</i>
<b>OASIS</b>	- <i>Organization for the Advancement of Structured Information Standards</i>
<b>ODF</b>	- <i>Open Document Format for Office Applications</i>
<b>ODT</b>	- <i>Open Document Text</i>
<b>OOo</b>	- OpenOffice.org
<b>ood</b>	- OpenOffice.org Daemon
<b>PDF</b>	- <i>Portable Document Format</i>
<b>PHP</b>	- <i>PHP Hypertext Preprocessor</i>
<b>PIM</b>	- <i>Personal Information Management</i>
<b>SGC</b>	- Sistemas de Gerenciamento de Conteúdo
<b>SGML</b>	- Linguagem Padronizada de Marcação Genérica

**SVN** – *SubVersioN*

**UNO** - *Universal Network Objects*

**W3C** - *World Wide Web Consortium*

**WAP** - *Wireless Application Protocol*

**WWW** - *Word Wide Web*

**XML** - *eXtensible Markup Language*

**ZODB** - *Zope Object Data Base*

**ZPL** - *Zope Public Lincese*

**ZTP** - *Zope Page Templates*

## SUMÁRIO

LISTA DE FIGURAS.....	VII
LISTA DE ABREVIATURAS E SIGLAS.....	IX
CAPÍTULO I - INTRODUÇÃO.....	1
CAPÍTULO II - REVISÃO BIBLIOGRÁFICA.....	3
2.1 Gestão do Conhecimento.....	3
2.1.1 Distinção entre conhecimento, dados e informação.....	4
2.2 Sistema de Gerenciamento de Conteúdo.....	5
2.3 Enterprise Content Management.....	5
2.4 Ensino à Distância.....	6
2.5 e-Learning.....	7
2.6 Objetos de Aprendizagem (learning objects).....	7
CAPÍTULO III - TECNOLOGIAS UTILIZADAS.....	9
3.1 Web Application .....	9
3.2 Zope.....	9
3.3 Plone .....	11
3.4 OpenDocument.....	11
3.4.1 Estrutura do OpenDocument.....	13
3.5 OpenOffice.org.....	14
3.5.1 PyUno.....	14
3.5.2 OpenOffice.org Daemon – oood.....	17
3.6 XML - eXtensible Markup Language.....	18
CAPÍTULO IV - SOFTWARE DESENVOLVIDO.....	19
4.1 GranulateO3Tool .....	19
4.1.1 Detalhes da Implementação.....	21
4.1.2 Identificação de Grãos.....	24
4.1.3 docBuilderO3.....	29
4.1.4 docBuilderRender.....	30
4.2 GrainCollector.....	31

4.3 GrainCart.....	33
4.4 Instalação.....	33
CAPÍTULO V - ESTUDO DE CASO.....	35
5.1 A Solução .....	35
5.1.1 Armazenamento.....	37
5.1.2 Processo de Granularização.....	38
5.2 Utilização .....	39
5.3 Geração de Conteúdo.....	43
CAPÍTULO VI - CONCLUSÃO.....	45
6.1 Estudos Futuros .....	45
REFERÊNCIAS.....	47

## CAPÍTULO I

### INTRODUÇÃO

Este trabalho é fruto de uma linha de pesquisa em *Enterprise Content Management* (ECM), realizada no Núcleo de Pesquisa em Sistemas de Informação – NSI, CEFET Campos, cujo objetivo é investigar os conceitos, abordagens e tecnologias para tratamento da informação semi e não-estruturada, propor métodos de modelagem e implantação de estruturas de ECM, investigar e desenvolver soluções de *Groupware*, *Document Engineering* (Engenharia de Documentos), Gestão de Conteúdo, *Information Retrieval* (Recuperação da Informação), Metadados, XML, *Personal Information Management* (PIM) e *Intranets*.

A ferramenta de granularização de conteúdo desenvolvida neste trabalho faz parte da solução em ECM desenvolvida no NSI nomeada NSI<sup>2</sup>, acrônimo de Núcleo de Pesquisa em Sistemas de Informação (NSI) e *Non-Structured Information*.

O NSI<sup>2</sup> é uma solução de ECM em software livre que inclui todo o ferramental de Gestão de Conteúdo, *Groupware*, PIM, Granularização de Documentos e Objetos de Aprendizagem. Altamente escalável, baseada no servidor de aplicações WEB Zope, no servidor de indexação NXLucene e Sistemas de Arquivos Distribuídos.

Está em uso no CEFET Campos e base para os portais do Sistema de Informações da Educação Profissional e Tecnológica (SIEP), sendo eles, Biblioteca Digital da EPT, Centro de Documentação Digital da EPT e Observatório Nacional da EPT.

O SIEP é uma iniciativa da Secretaria de Educação Profissional e Tecnológica (Setec/MEC), com apoio do Instituto Nacional de Estudos e Pesquisas Educacionais (Inep/MEC) e da Coordenação de Informática e Telecomunicação (Ceinf/MEC).

A motivação para o desenvolvimento da ferramenta apresentada neste trabalho surge diante da necessidade de atender um dos requisitos dos modernos aplicativos baseados em

conteúdo, onde se faz necessário o uso de ferramentas que possibilitem a reusabilidade de um conteúdo, seja ele num todo ou em partes.

A idéia de reusabilidade vem de encontro ao conceito de objetos de aprendizagem que pode-se definir como recursos reutilizáveis para o apoio do aprendizado. Segundo (RIBEIRO; BRAGA; PIMENTEL, 2006) os objetos de aprendizagem permitem uma grande flexibilidade, através de granularização do conteúdo. Isto permite que objetos educacionais sejam montados a partir de outros pré-existentes. Para (OLIVEIRA; BLANCO, 2003) um objeto de aprendizagem é então um “grão” de conteúdo autônomo que, associado a certos dados chamados metadados, poderá ser reutilizado em contextos e situações diferentes, permitindo a sua personalização em função de diferentes objetivos de aprendizagem e de diferentes perfis de estudantes. Esta granularização permite desdobrar o conteúdo relativo a um assunto em vários tópicos que podem ser recombinaados em diferentes recursos pedagógicos.

A granularização de documentos, possibilita que pedaços de informações se tornem conhecimento em diferentes contextos, retirando a visão do contexto global onde se encontra a referente informação. Este processo de granularização demonstra o valor que uma informação tem isolada abstraindo-se do ambiente onde ela está inserida.

Portanto este trabalho tem como objetivo demonstrar o desenvolvimento de uma ferramenta que utiliza-se de recursos computacionais para tratamento de informação semi-estruturada, capaz de efetuar a granularização de documentos, identificando grãos ou fragmentos de texto, sendo estes tabelas, imagens, parágrafos, fórmulas entre outros.

No capítulo II deste trabalho apresentamos uma revisão bibliográfica para fundamentar o estudo realizado. O capítulo III demonstra as tecnologias empregada para o desenvolvimento da solução. O software desenvolvido é apresentado no capítulo IV, e para demonstrar sua utilização foi realizado um estudo de caso em torno da criação de um repositório de objetos de aprendizagem para suporte ao Ensino à Distância descrito no capítulo V.

Este trabalho conclui-se no capítulo VI, onde se apresenta as conclusões tiradas acerca deste trabalho e as propostas de trabalhos futuros.



## **CAPÍTULO II**

### **REVISÃO BIBLIOGRÁFICA**

Este trabalho abordará o uso de uma ferramenta desenvolvida para o auxílio do tratamento da informação semi-estruturada aplicando-a no estudo de caso de um ambiente de Ensino à Distância, para tal este capítulo demonstra conceitos entorno da Gestão do Conhecimento, Sistemas de Gestão de Conteúdo, *Enterprise Content Management*, Ensino à Distância, *e-Learning* e Objetos de Aprendizagem (*learning objects*) dando fundamento as bases do estudo aqui demonstrado.

#### **2.1 Gestão do Conhecimento**

SALIM e SABBAG (2007) definem gestão do conhecimento como “ um processo sistemático, articulado e intencional, apoiado na geração, codificação, disseminação e apropriação de conhecimentos, com o propósito de atingir a excelência organizacional ”. Esse conceito cria rotinas e sistemas para que o conhecimento adquirido num determinado ambiente se expanda e seja compartilhado.

Segundo SILVA (2005) a busca pela obtenção do conhecimento, além da informação e da prática, mostra o fator humano e sua interação com o ambiente fundamental no enriquecimento e manifestação dos conhecimentos tácitos e explícitos dos membros da organização, inserindo as pessoas em interação produtiva, tornando-as parcerias na socialização, combinação, compartilhamento e apropriação de conhecimento produzido em equipes.

Ao se falar de gestão do conhecimento é importante que se tenha distinção dos

conceitos acerca de conhecimento, dados e informação.

### 2.1.1 Distinção entre conhecimento, dados e informação

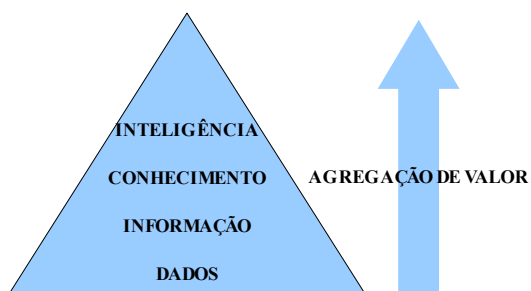
Vários autores buscam destacar a diferença entre dados, informação e conhecimento, porém não há um consenso quanto à diferenciação ou definição entre tais conceitos.

Para COSTI et al.(2001 apud DRUCKER 1998) conhecimento “é a informação eficaz em ação, focalizada em resultados”. Segundo COSTI et al.(2001 apud ROBREDO 1998) pode ser entendido como o resultado da agregação de valor à informação. COSTI et al.(2001 apud NONAKA) diz que conhecimento pode ser compreendido como tácito, quando inerente às pessoas que compõem as organizações, e explícito, quando agregado ou ligado aos produtos e serviços da organização.

Informação, segundo COSTI et al.(2001 apud DRUCKER), “é conjunto de dados que agrupados em função de uma determinada lógica, estabelecem o sentido ou uma ação”. Para COSTI et al.(2001 apud ROBREDO), “é um conjunto de dados agregados de um valor. Também pode ser compreendida como o resultado do ordenamento dos dados que proporciona a inferência de valor para a geração do conhecimento sistematizado com vistas à formação de conhecimento e o desenvolvimento de inteligência”.

Dados são definidos por SETZER (1999) como uma seqüência de símbolos quantificados ou quantificáveis. Também entende-se dados como um fenômeno qualquer desprovido de um significado, no momento em que o dado é contextualizado em um determinado nível de abstração passa a ser identificado como uma informação (BORGES; 2006).

A Figura 1 apresenta segundo ROBREDO (1998) a pirâmide de agregação de valor à informação, que contempla desde o dado até a inteligência.



*Figura 1: Pirâmide da agregação do valor à informação*

## **2.2 Sistema de Gerenciamento de Conteúdo**

Os Sistemas de Gerenciamento de Conteúdo (SGC) ou do inglês *Content Management Systems* (CMS) facilitam o gerenciamento e organização de documentos, imagens, mídias visuais, etc, no mundo virtual, passando assim a economizar e dinamizar a troca e disseminação de informações apoiando uma gestão do conhecimento eficaz. O objetivo é estruturar e facilitar a criação, administração, distribuição, publicação e disponibilidade da informação.

Segundo SVARRE (2006) as características de um CMS variam, mas a maioria incluem publicações baseadas na WEB, gerenciamento de formato, controle de revisão, indexação, busca e recuperação de informação.

Um CMS permite que se tenha total autonomia sobre o conteúdo e evolução da sua presença no sistema. Dispensa a assistência de terceiros ou empresas especializadas para manutenções de rotina. Segundo SVARRE (2006) nem mesmo é preciso um funcionário dedicado, pois cada membro da equipe poderá gerenciar o seu próprio conteúdo, diluindo os custos com recursos humanos. A habilidade necessária para trabalhar com um sistema de gerenciamento de conteúdo não vai muito além dos conhecimentos necessários para editar um texto no processador de textos.

Um sistema de gerenciamento de conteúdo reduz custos e ajuda a suplantando barreiras potenciais à comunicação WEB reduzindo o custo da criação, contribuição e manutenção de conteúdo.

## **2.3 Enterprise Content Management**

*Enterprise Content Management* (ECM) é uma tecnologia usada para capturar, gerenciar, armazenar, e disponibilizar conteúdo e documentos relacionados com o processo organizacional e integra o gerenciamento da informação estruturada, semi-estruturada e não estruturada. Todavia esta definição inclui documentos e informações em qualquer formato: de

sistemas legados, textos, páginas WEB, digitalizados, PDF, gráficos, vídeos, áudio, XML, aplicações WAP e etc.

O ECM é muitas das vezes confundido com uma série de outros tópicos entre eles pode-se citar o Gerenciamento do Conhecimento, Software de Gerenciamento de Conteúdo e Portais WEB. Entretanto ele representa uma nova área focado no gerenciamento de conteúdo textual e multimídia.

Segundo CARVALHO (2007) ECM é um tópico emergente que vem atraindo novos interessados tanto na área acadêmica quanto na profissional, incluindo fóruns profissionais como *AIIM International*<sup>1</sup>, que se auto-define como “a associação ECM”.

## **2.4 Ensino à Distância**

Para MORAN (2007) educação à distância e o processo de ensino-aprendizagem, onde alunos e professores estão separados por tempo ou localização, mediados por tecnologias. A expressão “ensino à distância” dá ênfase ao papel do professor como agente que ensina à distância, já o termo “educação” nos remete a uma visão mais abrangente.

Na educação à distância encontra-se possibilidades de combinar soluções pedagógicas adaptadas individualmente às peculiaridades da organização, às necessidades de cada momento. Encontra-se possibilidades centradas nas tecnologias *on-line*, textual, hipertextual, multimídia, podendo ministrar aulas ao vivo a distância por tele ou videoconferência.

Segundo MORAN (2007) em seu texto sobre Educação inovadora presencial e a distância, afirma que “o desafio inovador em EaD é superar o "conteudismo" e criar ambientes ricos de aprendizagem”.

Para (RIBEIRO; BRAGA; PIMENTEL, 2006) diversos recursos já foram utilizados em educação à distância, que tem evoluído de acordo com os avanços tecnológicos. A cada “ciclo tecnológico” houve o desenvolvimento de metodologias para o ensino à distância. É de comum acordo que o ciclo mais revolucionário está na utilização de tecnologias WEB, surgindo dessa forma o *e-learning*. Neste contexto, essas tecnologias propiciam a utilização de Objetos de Aprendizagem, como parte do material pedagógico de um curso.

---

<sup>1</sup> <http://www.aiim.org>

## 2.5 *e-Learning*

O *e-Learning* traz uma gama enorme de possibilidades para a difusão do conhecimento e da informação e tornou-se um novo mundo para a distribuição e o compartilhamento de conhecimento, tornado-se um canal para a democratização do saber propiciando que o conhecimento esteja disponível a qualquer tempo, hora ou lugar. Segundo (RIBEIRO; BRAGA; PIMENTEL, 2006) a chave para essa tecnologia de ensino está na interatividade e na possibilidade de oferecer ao aluno uma aprendizagem colaborativa e significativa.

Novas tecnologias foram desenvolvidas para suportar o processo de ensino/aprendizagem proposto pelo *e-Learning* dentre ele podemos citar os LMS's (*Learning Management System*), sistemas de gestão de ensino e aprendizagem na WEB. Softwares projetados para atuarem como salas de aula virtuais, gerando várias possibilidades de interações entre os seus participantes.

Com a interatividade conseguida através da WEB e os ambientes de gestão tem se um meio de comunicação entre aprendizes, orientadores capaz de propiciar uma interação em vários níveis, tais interações são descritas sendo:

- Aprendiz/Orientador;
- Aprendiz/Conteúdo;
- Aprendiz/Aprendiz;
- Aprendiz/Ambiente.

## 2.6 Objetos de Aprendizagem (*learning objects*)

A educação a distância se baseia, basicamente em objetos de aprendizagem (*learning objects*). Segundo MUZIO (2001, p.2), atualmente existem diversas definições diferentes sobre o termo objeto de aprendizagem, e muitos outros termos são utilizados. Isto sempre

resulta em confusão e dificuldade de comunicação, o que não surpreende devido a esse campo de estudo ser novo.

Para BECK(2002,p.1) (apud David A. Wiley) objeto de aprendizagem é qualquer recurso digital que pode ser reutilizado para o suporte ao aprendizado. A principal idéia do “objeto de aprendizagem” é quebrar o conteúdo educacional em pequenos pedaços que possam ser reutilizados em diferentes ambientes de aprendizagem, em um espírito de programação orientada a objetos. Outra definição é mostrada por BECK (apud IEEE) na qual se diz que objeto de aprendizagem é qualquer entidade, digital ou não digital, que possa ser usada, reutilizada ou referenciada durante o uso de tecnologias que suportem ensino.

Um consenso entre as definições é de que a principal característica dos objetos de aprendizagem é sua reusabilidade, que é posta em prática através de repositórios, que armazenam os objetos logicamente, permitindo serem localizados a partir da busca por temas, por nível de dificuldade, por autor ou por relação com outros objetos.

Para que possa ocorrer a recuperação e reutilização o objeto de aprendizagem deve ser devidamente indexado e armazenado. Contudo o processo de indexação do objeto, ou seja, o preenchimento dos metadados é um gargalo no desenvolvimento dos objetos de aprendizagem, pois esse processo se torna trabalhoso e demanda muito tempo. Para (RIBEIRO; BRAGA; PIMENTEL, 2006) o aluno pode buscar exatamente o que precisa, baseado em informações exteriores, sem necessidade de conhecer o todo de um objeto para saber se seu conteúdo lhe pode ser útil, para tal se faz necessário o uso de metadados padronizados, como por exemplo o padrão internacional LOM (IEEE, 2002), pois em diversos casos criadores e indexadores de objetos de aprendizagem interpretam de forma diferentes os valores fornecidos para à indexação, o que leva à metadados incompletos, com valores ambíguos ou semanticamente inconsistentes, o que acaba por prejudicar a recuperação e, conseqüentemente, a reutilização.

## CAPÍTULO III

### TECNOLOGIAS UTILIZADAS

Este capítulo aborda as tecnologias, padrões e ferramentas de código aberto utilizados no desenvolvimento do GranulateO3Tool.

#### 3.1 Web Application

Em engenharia de software um Web Application ou *webapp* é um aplicativo que pode ser acessado via Web ou por uma rede, tal como a Internet ou uma Intranet.

Para MONNERAT e LEAL (2004) a Word Wide Web (WWW) deixou de ser uma coleção de páginas estáticas e passou a prover também serviços, desses podemos citar as bases de dados, mecanismos de busca, portais entre outros. Desta forma considera-se alguns desses serviços Web Application, pois se faz o uso de um navegador apenas para serem acessados e utilizados.

Para suprir a necessidade de rapidez e facilidade na criação de Web Application surgiram tecnologias alternativas como JSP, ASP, PHP, ColdFusion.

#### 3.2 Zope

Zope é um acrônimo para *Z Object Publishing Environment*, foi desenvolvido pela

Digital Creations Inc., atualmente Zope Corporation<sup>2</sup>. De acordo com FERRI (2002) o Zope tem aproximadamente 85% do seu código escrito em Python, e alguns módulos críticos onde se necessita de performance o seu código foi convertido para C++ . Ele é distribuído sobre a ZPL (*Zope Public License*), uma licença *Open Source* que vem de acordo com a GPL (*General Public License*), por tal motivo seu código pode ser estudo e alterado. Pode-se executá-lo em plataformas Unix, NT, MAC e MainFrames.

O *DevGuide* do Zope, segundo FERRI (2002 apud Latteier 01), define o Zope como um *framework Open Source* para aplicações Web, essa definição se torna muito restrita, tendo em vista que o Zope pode ser mais amplamente definido como é mostrado por FERRI (2002). Ele classifica o Zope sendo um ambiente para construção e gerenciamento de aplicações Web, focado no conteúdo e também um gerador de páginas Web dinâmico, que interpreta os fragmentos de códigos Zope e HTML em seus *templates* e gera um HTML compreensível para qualquer navegador.

O Zope possui duas linguagens lógicas próprias, a DTML e a ZPT. A DTML é um acrônimo para "*Document Template Markup Language*" (Linguagem de Marcação para Modelos de Documentos), é uma linguagem de *scripts*, ou seja, o código não é compilado e sim interpretado, oriunda da família SGML (FERRI 2001). A ZPT (*Zope Page Templates*) ou Modelos de Páginas do Zope, é uma ferramenta para geração de página Web. Elas ajudam programadores e designers a produzir páginas Web dinâmicas em colaboração, para aplicações Zope (FERRI 2002 apud Latteier 02). Para lhe conferir maior poder e flexibilidade o Zope conta com o suporte a *scripts* em Python .

Também ocorre que o Zope possui seu próprio servidor Web, o Zserver (*Zope Server*), que dispensa o uso de qualquer outro servidor, mas não impede que ocorra a integração com outros servidores Web, como por exemplo o uso do servidor Web Apache.

Utiliza-se como base de dados própria o ZODB (*Zope Object Data Base*), que é um banco de dados transacional orientado à objetos, podendo também interagir com outros bancos de dados relacionais como Oracle, MySQL, PostgreSQL, entre outros.

O Zope conta com uma gama enorme de produtos, ou seja extensões, desenvolvida por sua ativa comunidade de desenvolvedores e mantida por seu site oficial<sup>3</sup>. Dentre esses produtos está o CMS Plone.

---

<sup>2</sup> [www.zope.com](http://www.zope.com)

<sup>3</sup> [www.zope.org](http://www.zope.org)



### 3.3 Plone

O Plone é um robusto Sistema Gerenciador de Conteúdo de código livre escrito em Python e trabalha sobre o Zope e o *framework* CMF (*Content Management Framework*).

O CMF de acordo com MONNERAT e LEAL (2004) é um aplicação que contém uma série de ferramentas para o Zope, ferramentas essas que formam um *framework* com as principais funções que um Sistema Gerenciador de Conteúdo necessita.

Segundo BURTON (2007), o Plone é um sistema fácil de configurar, extremamente flexível e prove um sistema de gerenciamento de conteúdo Web ideal para atividades em grupo, comunitárias, *Web sites*, *extranets* e *intranets*.

Ele conta com a função de *workflow*, segurança, funções pré-configuradas, um conjunto de tipos de conteúdo básicos, tais como pastas, *links*, arquivos, documentos, eventos, imagens e notícias, suporte a vários idiomas e uma extensa lista de produtos adicionais, suportada por uma comunidade ativa de desenvolvedores.

### 3.4 OpenDocument

OpenDocument ou ODF, abreviação de "OASIS *Open Document Format for Office Applications*", é um formato aberto para armazenar e intercambiar documentos de escritório como memorandos, relatórios, livros, planilhas, bases de dados, gráficos e apresentações.

Esse padrão foi desenvolvido pelo consórcio OASIS e é baseado no formato XML. O acesso a este padrão é público, o que possibilita a implementação em qualquer sistema, seja ele de código aberto ou não. A intenção do formato ODF é prover uma alternativa aos formatos proprietários de documentação, para que não se fique aprisionado a um único fornecedor. O ODF é um padrão ISO/IEC aprovado e lançado em maio de 2006 (ISO/IEC 26300), já anteriormente aprovado como um padrão OASIS em 1 de maio de 2005.

As Extensões de arquivos usadas pelos documentos do OpenDocument são:

- .odt para Processadores e Texto (*text*)
- .ods para Planilhas Eletrônicas (*spreadsheets*)
- .odp para Apresentações em Slides (*presentations*)
- .odb para Banco de Dados (*database*)
- .odg para Editor de imagens (*graphics*)
- .odi para Images (*image*)
- .odf para Equações Matemáticas (*formulae*)

As extensões específicas para *templates*, ou seja, documentos que servem como modelos são:

- .ott para Processadores De Texto (*text*)
- .ots para Planilhas Eletrônicas (*spreadsheets*)
- .otp para Apresentações em Slides (*presentations*)
- otg para Editor de imagens (*graphics*)

Atualmente o OpenDocument é compatível com um grande número de aplicativos, na Tabela 1 são descritos alguns aplicativos processadores de textos que suportam o formato ODT.

<b>Processadores de Texto – <i>Text documents</i> (.odt)</b>	
Aplicativo	Descrição
Abiword 2.4	<i>Free Software</i> , licenciado pela GNU <i>General Public License</i> . Suportado nos sistemas operacionais Linux, Mac OS X (PowerPC), Microsoft Windows, ReactOS, BeOS, AmigaOS 4.0 entre outros.
ajaxWrite	Processador de textos baseado na Web, pode ler e escrever arquivos no formato ODF. Endereço web: <a href="http://us.ajax13.com/en/ajaxwrite/">http://us.ajax13.com/en/ajaxwrite/</a>
Coventi Pages	Processador de textos baseado na Web e revisor de documentos colaborativos.
Google Docs	Um pacote de aplicativos para escritório do <i>Google</i> baseado em AJAX. Funciona totalmente <i>on-line</i> diretamente no <i>browser</i> .
IBM Lotus Symphony	Conjunto de aplicativos para criação, edição e compartilhamento de texto, planilhas e outros documentos.

Ichitaro (Japonês)	Processador de textos em japonês, lê e escreve no formato <i>OpenDocument</i> .
KWord	<i>Free Software</i> , membro do projeto KOffice desenvolvido pela K Desktop Environment.
Microsoft Word	Componente da suíte de escritório Microsoft Office. Não tem suporte nativo ao ODF, mas disponível através de <i>plugin Open Source</i> .
OpenOffice.org Writer	Componente da suíte de escritório OpenOffice.org. Tem suporte completo ao ODF a partir da versão 2.0.
TextEdit	Processador e editor de textos <i>Open Source</i> distribuído no Mac OS X da Apple.
TextMaker 2006	Processador de textos que pode ser adquirido separadamente ou na suíte de escritório SoftMaker. Trabalha nos Sistemas Operacionais Linux, Windows bem como em Pocket PCs, Handheld PCs, Windows CE.NET, FreeBSD e Sharp Zaurus.
Zoho Writer	Processador de textos <i>on-line</i> , pode ler e escrever no formato ODT

*Tabela 1: Aplicativos processadores de texto que suportam o formato ODT*

### 3.4.1 Estrutura do OpenDocument

Um arquivo *OpenDocument* básico consiste em um documento XML que contém `<office:document>` como elemento raiz. Os arquivos *OpenDocument* são compactados e contam com uma estrutura de arquivos e diretórios que separam os conteúdos, estilos, metadados e os arquivos binários. Os principais arquivos e diretórios são:

- **content.xml** - arquivo que armazena o conteúdo do documento criado pelo usuário;
- **meta.xml** - armazena os metadados do documento, ou seja, informações como autor, data de modificação, quantidade de palavras, etc;
- **styles.xml** - contém estilos para o documento, por exemplo, formatações específicas para parágrafos e listas;
- **Pictures (pasta)** - diretório que armazena as imagens do documento.

### 3.5 OpenOffice.org

O OpenOffice.org é uma suíte de escritório e um projeto de código aberto, multiplataforma e com suporte a vários idiomas, compatível com a maioria das suítes de escritórios.

O OpenOffice.org é baseado em uma antiga versão do StarOffice. O StarOffice 5.1 foi adquirido pela Sun Microsystems em Agosto de 1999. O código fonte da suíte foi liberado dando início a um projeto de desenvolvimento de um software de código aberto em 13 de outubro de 2000, o OpenOffice.org. O principal objetivo era fornecer uma alternativa de baixo custo, de alta qualidade e de código aberto.

Para MINNETO (2004) o projeto OpenOffice.org possui uma característica muito útil e pouco utilizada que é a capacidade de integrar seu funcionamento com outros aplicativos. Tal funcionalidade é empregada através do UNO (*Universal Network Objects*), que é um modelo de componentes do OpenOffice.org.

O UNO oferece interoperabilidade entre diferentes linguagens de programação, modelos de objetos diferentes, arquiteturas e processos diferentes, em uma rede local ou mesmo através da internet. Os componentes podem ser utilizados e acessados por qualquer linguagem de programação que possua acesso aos *bindings* do UNO, atualmente C, C++ , Java e Python possuem essas ferramentas

#### 3.5.1 PyUno

Na linguagem de programação Python o acesso ao UNO é possível utilizando-se o Python-Uno (PyUno), já incluso como padrão na instalação do OpenOffice.org desde a versão 1.1RC4. O conector do PyUno permite usar a API padrão do OpenOffice.org e pode ser utilizado de três modos diferentes:

1. Utilizado com o *framework* de *scripts* do OpenOffice.org;

O OpenOffice suporta a partir da versão 2.0, mais precisamente a partir da versão OOo 1.9.m79 ou OOo 2.0 beta, a execução de *scripts* no seu *framework*. A versão é limitada ao núcleo do *framework*, significando que a execução e a atribuição de macros é bem suportada, mas a edição e a depuração de macros não é integrada a interface do OpenOffice.org. A Figura 2 demonstra a interface de execução de macros no OpenOffice.org.

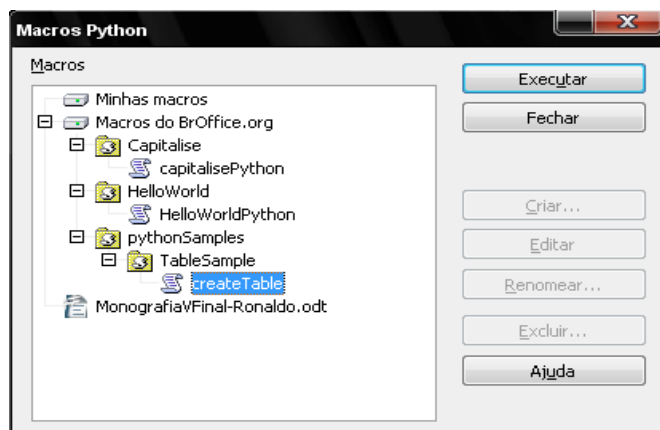


Figura 2: Interface de execução de scripts Python OpenOffice.org

As macros em Python são inseridas no diretório *Scripts*, localizado na instalação do OpenOffice.org. O diretório *Scripts* contém uma estrutura de pastas onde são organizados os *scripts* por linguagem de programação.

## 2. Dentro de um executável Python ;

Pode se criar códigos em Python e executá-los fora do processo do OpenOffice.org, como é exposto na Figura 3, onde se demonstra a utilização de um *socket* OpenOffice.org para execução de tarefas relativas as APIs do OpenOffice.org. Esta forma de utilização torna a execução dos *scripts* demorada, pois um interprocesso OpenOffice.org abre um *socket* para receber as requisições do Python *script* que implementa um código para fazer uma *interprocessbridge* entre o *python code* e o OpenOffice.org

## Common python scripts using PyUNO

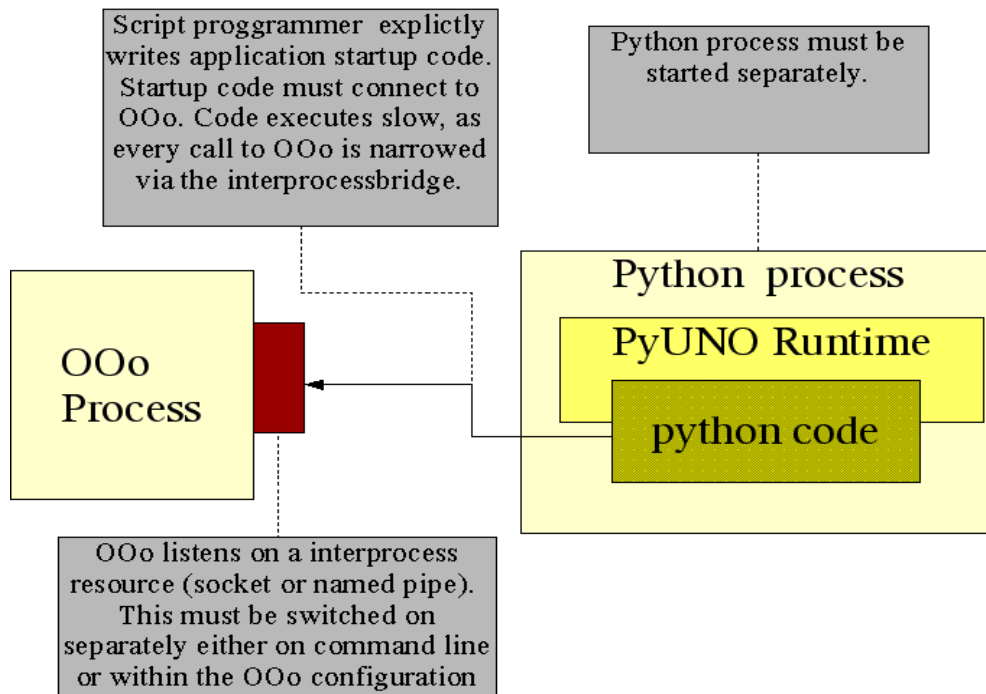


Figura 3: Scripts Python utilizando o PyUno - Fonte: OPENOFFICE.ORG

### 3. Embutido ao processo do OpenOffice.org;

Nesta forma de utilização pode-se criar *scripts* Python como componentes do UNO, com isso há um ganho de desempenho, pois não há necessidade de um *interprocessbridge* para efetuar a conexão entre o processo OpenOffice.org e o Python *code*. Esta forma de utilização possibilita a captura de eventos na interface do OpenOffice.org. Ver Figura 2.

## Python scripts as UNO components

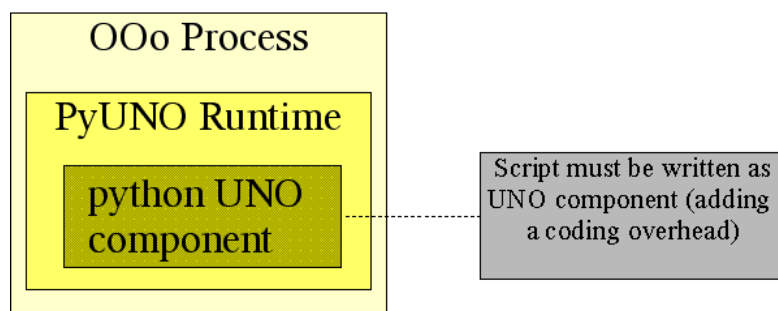


Figura 4: Scripts Python como Componentes do UNO -  
Fonte: OPENOFFICE.ORG

### 3.5.2 OpenOffice.org Daemon – oood

O nome “oood” é uma abreviatura para OpenOffice Daemon, um servidor de chamada de procedimento remota, que implementa o protocolo XML-RPC, escrito em Python desenvolvido pela empresa francesa NEXEDI<sup>4</sup>, desenvolvedora do sistema de *Enterprise Resource Planning* de código aberto, ERP5<sup>5</sup>.

O oood é um servidor capaz de efetuar conversão de tipo entre documentos de escritório, como por exemplo conversão entre os formatos MSOffice e OpenOffice, que possui as seguintes características (ERP5) :

- Suporta a geração de documentos em formato PDF e HTML;
- possibilita a edição e extração de metadados em documentos;
- capaz de fazer todas as operações que o OpenOffice.org faz, com a limitação no que se refere as opções de salvamento de arquivos, ele sempre faz uso de valores padrão.

O processo do OpenOffice.org é inicializado através de um *script* de inicialização que o faz em *background* e usa *display* virtual. O servidor fica aguardando por uma requisição e cria um “*pool*” para evitar gargalo nas requisições.

Para que o tempo de resposta seja aceitável o *daemon* cria múltiplas instâncias do OpenOffice.org para atender a demanda nas requisições, ou seja ocorre a formação de um “Grid OpenOffice”. Ver Figura 5.

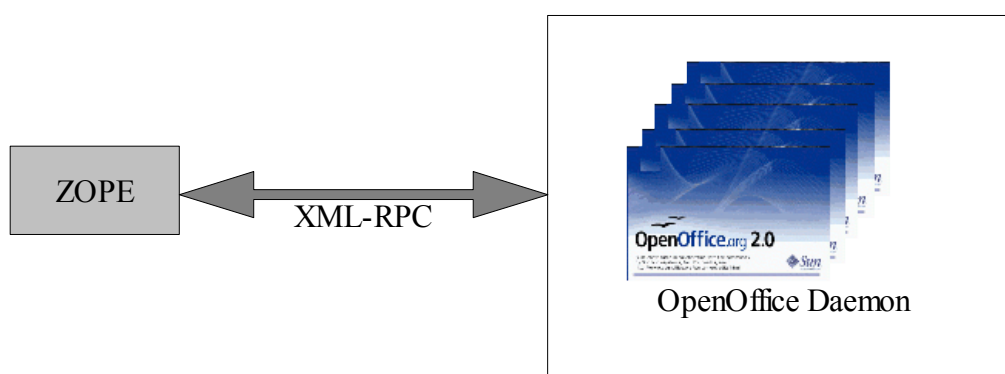


Figura 5: Esquema de comunicação servidor ZOPE e oood

4 [www.nexedi.com](http://www.nexedi.com)

5 [www.erp5.org](http://www.erp5.org)

### 3.6 XML - *eXtensible Markup Language*

O XML é uma recomendação da W3C para geração de linguagens de marcação para necessidade especiais. A W3C (*World Wide Web Consortium*) é um consórcio de empresas de tecnologia fundada por Tim Bernes-Lee em 1994 para otimizar o uso da Web, via protocolos comuns e fóruns abertos que promovem sua evolução e asseguram a sua interoperabilidade. O propósito principal do XML é facilitar o compartilhamento de informações através da Internet. O XML é um subtipo da Linguagem Padronizada de Marcação Genérica - SGML, que tem a capacidade de descrever diversos tipos de dados.

A Figura 6 demonstra um exemplo de XML criado para intercambiar dados sobre alunos de uma determinada instituição.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<alunos>
  <aluno>
    <nome>João da Silva</nome>
    <email>joaodasilva@ygh.com</email>
    <nascimento>16/12/1989</nascimento>
    <turma>1M</turma>
    <serie>1 EM</serie>
  </aluno>
</alunos>
```

*Figura 6: Exemplo de arquivo XML*

O XML é considerado um bom formato para a criação de documentos com dados organizados de forma hierárquica, como são organizados por exemplos documentos de texto formatados, imagens vetoriais ou bancos de dados. Sua portabilidade pode ser aplicada, por exemplo, na exportação entre bancos de dados, onde um determinado banco de dados escreve um arquivo XML com seus dados e um banco de dados distinto faz uso dos mesmos.



## CAPÍTULO IV

### SOFTWARE DESENVOLVIDO

#### 4.1 GranulateO3Tool

O GranulateO3Tool é uma extensão para o CMS Plone que provê a retirada de grãos ou fragmentos de arquivos de texto, sendo estes grãos tabelas e imagens. Outros elementos vinculados ao documento também são retirados como o sumário e o *thumbnails*, uma imagem em miniaturas representando a primeira página do documento.

A ferramenta possibilita a retirada automática e inserção de metadados em documentos, como por exemplo autor e palavra-chave. Conta com serviços de conversão de documentos, criação de HTML e PDF e um conjunto de ferramentas de apoio a geração de conteúdo denominadas docBuilderO3, docBuilderRender, GrainCollector e GrainCart que permitem ao usuário montar documentos a partir de grãos de outros documentos.

O GranulateO3Tool requer um servidor OpenOffice Daemon (ood), pois as funções de retirada e inserção de metadados, conversão de arquivos e criação de HTML e PDF é suportada pelo mesmo.

Para que ocorra a integração com o OpenOffice Daemon (ood), se implementou uma interface de configuração demonstrada na Figura 7, onde se indica a porta de comunicação com o servidor e o seu endereço.

## GranulateO3Tool Settings

[▲ Up to Site Setup](#)

**GranulateO3Tool settings for this site.**

GranulateO3Tool details

**Port** ■

8008

**Host** ■

10.10.6.6

save

*Figura 7: Interface de configuração GranulateO3Tool*

A inserção e retirada de metadados pode se efetuada nos seguintes formatos de arquivos:

- Open Document Text – odt;
- Open Document Spreadsheet - ods;
- Open Document Presentation – odp;
- OpenOffice.org Writer 1.0 – sxw;
- OpenOffice.org Calc 1.0 – sxc;
- OpenOffice.org Impress 1.0 – sxi;
- StarWriter versão 5.0/4.0/3.0 – sdw;
- StarCalc versão 5.0/4.0/3.0 – sdc;
- StarImpress versão 4.0/5.0 – sdd;
- StarDraw versão 3.0 – sda;
- StarDraw versão 5.0 – sdd;
- Microsoft Word 97/2000/XP;
- Microsoft Excel 97/2000/XP;
- Microsoft PowerPoint 97/2000/XP;

A granularização de tabelas e imagens pode ser realizada nos seguintes formatos de arquivos:

- Open Document Text – odt;
- Open Document Presentation – odp;
- Microsoft Word 97/2000/XP;

- Microsoft PowerPoint 97/2000/XP;

O formato *Portable Document Format* PDF suporta somente a granularização de imagem, pois a recuperação de tabelas, atualmente, consiste no desenvolvimento de uma heurística apropriada que ainda assim tem alto custo computacional ou implica na utilização de software proprietário para conversão.

#### 4.1.1 Detalhes da Implementação

A Figura 8 demonstra o *Deployment Diagram* do GranulateO3Tool onde são mostrados os relacionamentos físicos e lógicos entre os componentes de hardware e software do sistema. Este diagrama é composto por três instâncias de nós, a primeira representa uma *Workstation*, a segunda representa um *Web Server* e o terceiro representa um *Server OpenOffice Daemon*.

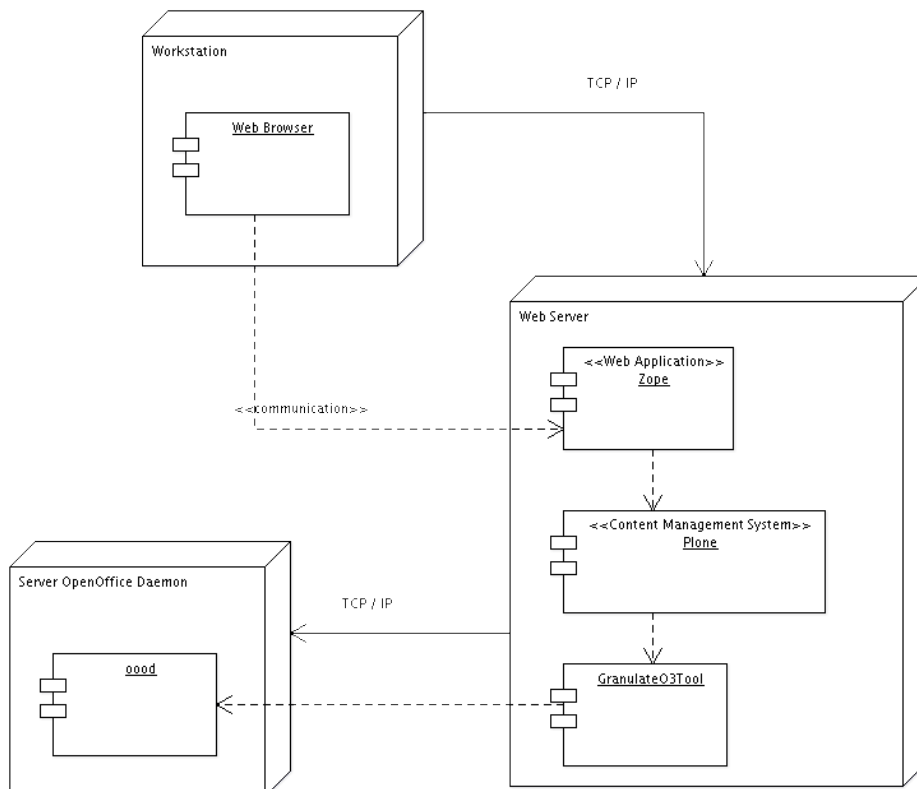


Figura 8: Deployment Diagram GranulateO3Tool

O GranulateO3Tool tem um comportamento de serviço para o Plone como pode ser visto na Figura 9, onde se demonstra o Diagrama de Classe do GranulateO3Tool.

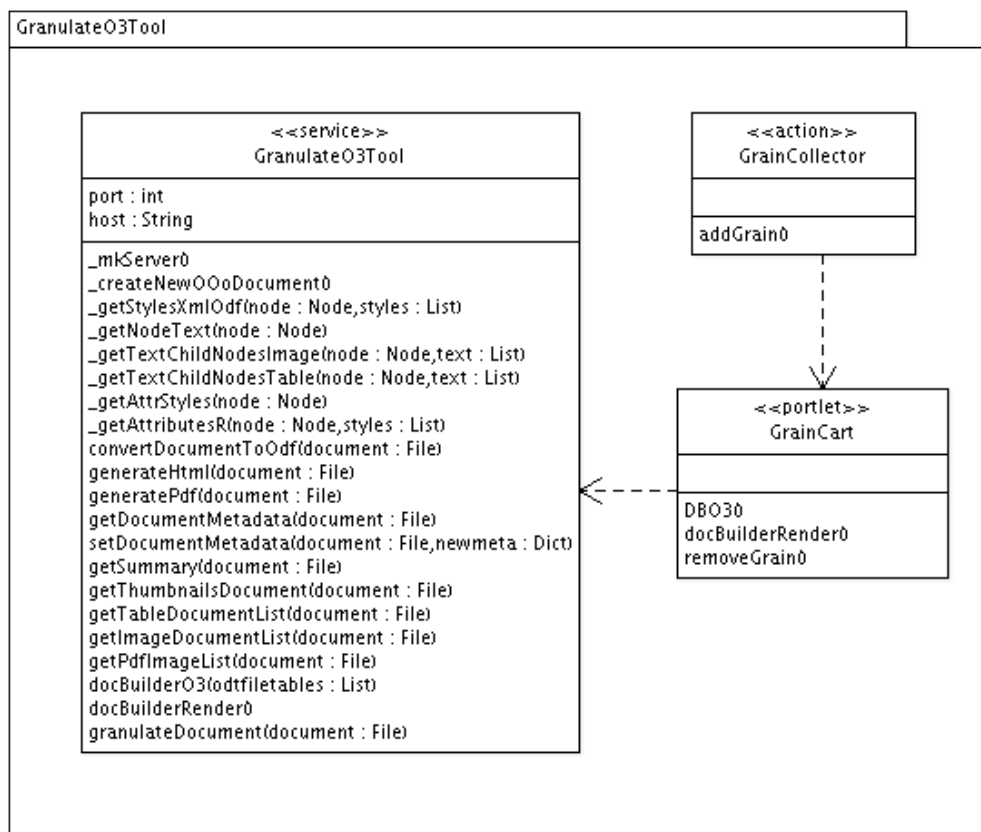


Figura 9: Diagrama de Classe GranulateO3Tool

A classe GranulateO3Tool contém os atributos *port* e *host* que são usados como parâmetro para estabelecer a comunicação com o servidor OpenOffice Daemon e conta com métodos públicos e privados que implementam as funcionalidades gerais da ferramenta.

O pacote GranulateO3Tool implementa duas ferramentas anteriormente mencionadas que auxiliam no processo de coleta e gerenciamento dos grãos de documentos, a *action* GrainCollector e o *portlet* GrainCart.

Todo processo de identificação e retirada dos grãos efetuada pelo GranulateO3Tool se baseia no processamento de arquivos XML que são a base do padrão OpenDocument ODF, diante deste cenário existe a necessidade de efetuar a conversão dos arquivos cujo padrão não se baseia no OpenDocument. Tal conversão é efetuada pelo método **convertDocumentToOdf** demonstrada no trecho de código na Figura 10, onde se estabelece uma comunicação com o servidor OpenOffice Daemon invocando seu método de conversão de arquivos.

```

225     def convertDocumentToOdf(self, document=None):
226         """
227         Convert one document to Open Document Format (odf)
228         """
229         sp = self._mkServer()
230         data=str(document)
231         res = sp.convert(document.filename, base64.encodestring(data))
232         if res[0]==200:
233             file=StringIO(base64.decodestring(res[1]['data']))
234             return file
235         else:
236             # logar em event.log os erros
237             return None

```

Figura 10: Trecho de código função de conversão de arquivos

A linha 225 da Figura 10 demonstra a assinatura do método onde é esperado como parâmetro um documento de texto. Na linha 229 é invocado um método privado, `_mkServer`, para obter a conexão com OpenOffice Daemon, que retorna um objeto com as funcionalidades suportadas pelo mesmo.

Na linha 230 atribui-se a variável `data` uma *string* que representa o conteúdo do arquivo de texto obtido através da função `str` definida pelo Python. Na linha 231, o método `convert` do servidor OpenOffice Daemon é invocado, passando como parâmetro o nome e o conteúdo do arquivo capturado através da variável `data`, cujo conteúdo é codificado para Base64, um método de codificação de dados para transferência na Internet .

O retorno da operação de conversão efetuada pelo OpenOffice Daemon é representada por um lista, que descreve em seu primeiro elemento um valor que indica o *status* da ação, os valores de retorno são os seguintes:

- valor 200, operação realizada com sucesso;
- valor 401, método requisitado não é suportado;
- valor 402, o documento não pode ser processado;
- valor 403, *timeout* enquanto documento é processado;
- valor 404, requisição mal formada,
- valor 501, o servidor está reiniciando;
- valor 502, o *poll* está vazio, o servidor será reiniciado;
- valor 503, o servidor está fora de sincronia e será reiniciado;
- valor 504, erro não identificado.

O segundo elemento desta lista é um dicionário com os dados e metadados do

documento convertido, tendo como chaves os seguintes valores: *data*, *title*, *subject*, *keywords*, *description*, *MIMEType*, *reference*, *version*, *language*.

No intervalo das linhas 232 a 237, verifica se o primeiro valor da lista *res* corresponde a 200, caso seja verdadeiro é retornado uma variável *file* contendo um objeto do tipo *StringIO* com o conteúdo do arquivo decodificado. Caso a condição seja falsa é retornado *None*.

#### 4.1.2 Identificação de Grãos

Todo conteúdo textual de um arquivo OpenDocument ODF é contido em um arquivo XML denominado *content.xml*, que se encontra compactado em uma estrutura de arquivos, anteriormente citada no Capítulo III onde são descritas as tecnologias utilizadas. Os grãos são identificados a partir do processamento deste arquivo XML, no qual temos as tabelas definidas pela tag *table:table* e imagem *draw:image*. As imagens são retiradas pela função **getImageDocumentList** demonstrada na Figura 11.

```
459 def getImageDocumentList(self, document=None):
460     """
461     Get and create new Document with image
462     Document is instance from Gran(id,caption,content)
463     """
464     image_list=[]
465     if isinstance(document,StringIO):
466         filedoc = document
467     else:
468         filedoc = StringIO(str(document))
469     odf_file = zipfile.ZipFile(filedoc, 'r')
470     contents = odf_file.read('content.xml')
471     dom = parseString(contents)
472     tag_images = dom.getElementsByTagName('draw:image')
473     if len(tag_images):
474         for item in tag_images:
475             path=item.getAttribute('xlink:href')
476             if "Pictures" in path:
477                 name=path.replace("Pictures/","")
478                 if not name in [image.getId for image in image_list]:
479                     (sysname, ext)=os.path.splitext(name)
480                     parent = item.parentNode
481                     nChild = parent.nextSibling
482                     objGran = Grain()
483                     if nChild:
484                         text=[]
485                         caption = ''
486                         if nChild.nodeType is nChild.TEXT_NODE:
487                             text.append(nChild.data)
488                             for t in self._getTextChildNodesImage(nChild,text):
489                                 if t is not None: caption+=t
490                             objGran.setCaption(caption)
491                             imagefile=StringIO(odf_file.read(path))
492                             objGran.setId(name)
493                             objGran.setContent(imagefile)
494                             image_list.append(objGran)
495             dom.unlink()
496         odf_file.close()
497     if image_list:
498         return image_list
499     else:
500         return None
---
```

Figura 11: Método de retirada de imagem *getImageDocumentList*

A linha 459 do trecho de código da Figura 11 demonstra a assinatura do método **getImageDocumentList**, onde se espera como parâmetro um arquivo de texto padrão OpenDocument ODF ou uma instância de *StringIO* que represente o arquivo. No intervalo entre as linhas 465 a 468 é efetuado uma condição que verifica se o documento recebido como parâmetro através da variável *document* é uma instância de *StringIO*, caso seja verdadeiro a variável *filedoc* recebe o documento passado, caso contrário o documento é convertido para uma instância de *StringIO* e atribuído a *filedoc*.

Na linha 469 o arquivo contido na variável *filedoc* é descompactado utilizando a biblioteca *ZipFile* do Python e o conteúdo da operação é armazenado na variável *odf\_file*. Na linha 470 o arquivo *content.xml* contido em *odf\_file* é lido e o seu conteúdo XML é armazenado na variável *contents*. Na linha 471 é efetuado um *parse*, utilizando a biblioteca DOM (*xml.dom.minidom.parseString*), em *contents* e atribuído a variável *dom* o resultado da operação, que consiste em uma instância de *Document* (*xml.dom.minidom.Document*).

Na linha 472 o método **getElementsByTagName** é invocado para se capturar os elementos *draw:image*, que representam as imagens no padrão OpenDocument. Este método retorna uma lista de elementos *draw:image* que são armazenados na variável *tag\_images*. Na linha 473 um teste é realizado para verificar se *tag\_images* contém algum elemento, se existir algum elemento, uma interação é efetuada na lista no intervalo das linhas 474 a 494.

Todo elemento *draw:image* possui um atributo chamado *xlink:href* onde encontra-se o caminho para a imagem. Na linha 475 o valor deste atributo é capturado pelo método *getAttribute* e armazenado na variável *path*. Nas linhas 476 e 477 o nome da imagem é retirado do *path* e armazenado na variável *name* que é utilizada em seguida na linha 478 para verificar se a imagem capturada já existe na lista *image\_list*.

Nas linhas 480 e 481 é capturado na hierarquia do XML o próximo elemento de *draw:image* que consiste na legenda da imagem. A legenda é considerada todo corpo de texto ancorado a imagem. Entre as linhas 483 e 489 a legenda é retirada e armazenada na variável *caption*.

Na linha 482 um objeto **Grain** é instanciado em *objGran*, o objeto **Grain** é uma referência genérica para os grãos onde se encontra os atributos *id*, *caption* e *content*, que correspondem respectivamente a identificação, a legenda e o conteúdo da imagem.

As linhas 490, 491, 492 e 493 efetuam respectivamente o trabalho de inserir em

*objGran*, a legenda, o nome e o conteúdo da imagem armazenado na variável *imagefile*. A linha 494 armazena na lista *image\_list* o objeto **Grain** instanciado. Esta lista consiste no retorno do método, que é realizado no intervalo das linhas 497 a 500, onde uma condição é realizada para verificar se a lista contém algum objeto.

A identificação e retirada de imagens de arquivos PDF é implementada pelo método **getPdfImageList**, este método utiliza a biblioteca *xpdf-common* que provê a função *pdfimages*.

O processo de retirada das tabelas é realizado pelo método **getTableDocumentList**, o procedimento de leitura do arquivo *content.xml* é o mesmo anteriormente citado no método **getImageDocumentList**. O parser XML identifica as tabelas através da tag *table:table* e retorna uma lista de elementos tabela.

Cada tabela tem associada um ou mais estilos, ou seja a formatação da tabela, onde são definidas cores, fontes, tipo bordas, entre outros. Cada estilo é definido na tag *style:style*, para associar os estilos à tabela encontra-se um atributo *style-name* definido para cada elemento específico da tabela. Como exemplo pode-se observar o trecho de código XML retirado do arquivo *content.xml* representado nas Figuras 12 e 13.

```
- <office:automatic-styles>
- <style:style style:name="Tabela1" style:family="table">
  <style:table-properties style:width="16.999cm" table:align="margins"/>
</style:style>
- <style:style style:name="Tabela1.A" style:family="table-column">
  <style:table-column-properties style:column-width="8.498cm" style:rel-column-width="32767*"/>
</style:style>
- <style:style style:name="Tabela1.B" style:family="table-column">
  <style:table-column-properties style:column-width="8.5cm" style:rel-column-width="32768*"/>
</style:style>
+ <style:style style:name="Tabela1.A1" style:family="table-cell"></style:style>
- <style:style style:name="Tabela1.B1" style:family="table-cell">
  <style:table-cell-properties fo:padding="0.097cm" fo:border="0.002cm solid #000000"/>
</style:style>
+ <style:style style:name="Tabela1.A2" style:family="table-cell"></style:style>
- <style:style style:name="Tabela1.B2" style:family="table-cell">
  <style:table-cell-properties fo:padding="0.097cm" fo:border-left="0.002cm solid #000000" fo:border
fo:border-top="none" fo:border-bottom="0.002cm solid #000000"/>
</style:style>
- <style:style style:name="Table1" style:family="table">
  <style:table-properties style:width="16.999cm" table:align="margins" fo:keep-with-next="always"/>
</style:style>
```

Figura 12: Trecho de código XML representando os estilos do documento



```

- <table:table table:name="Table1" table:style-name="Table1">
  <table:table-column table:style-name="Table1.A"/>
  <table:table-column table:style-name="Table1.B"/>
- <table:table-row>
  - <table:table-cell table:style-name="Table1.A1" office:value-type="string">
    <text:p text:style-name="P19">Descrição </text:p>
  </table:table-cell>
  - <table:table-cell table:style-name="Table1.B1" office:value-type="string">
    <text:p text:style-name="P19">Data</text:p>
  </table:table-cell>
</table:table-row>
- <table:table-row>
  - <table:table-cell table:style-name="Table1.A2" office:value-type="string">
    <text:p text:style-name="P11">
      Busca de inovações no processo produtivo desenvolvidas por instituições de pesquisas.
    </text:p>
  </table:table-cell>
  - <table:table-cell table:style-name="Table1.B2" office:value-type="string">
    <text:p text:style-name="P18">Segundo semestre de 2008</text:p>
  </table:table-cell>
</table:table-row>

```

Figura 13: Trecho de código XML representando o elemento Tabela

A Figura 12 mostra o trecho do XML onde se localiza os estilos do documento, estando em foco o trecho que descreve o estilo de uma tabela.

Na Figura 13 é demonstrado o XML que constrói a tabela no ODT, este contém no elemento *table* um atributo *table:style-name* que define o estilo do elemento. Através deste atributo que se relaciona o estilo ao elemento.

A retirada do estilo é efetuada por duas funções demonstradas na Figura 14, elas trabalham identificando todos os estilos associados a um determinado elemento tabela.

```

215 security.declarePrivate('_getAttrStyles')
216 def _getAttrStyles(self, Node):
217     """
218         get Styles
219     """
220     if Node.attributes is not None:
221         for i in Node.attributes.keys():
222             if re.search("^.+|:style-name$",i):
223                 if Node.getAttribute(i):
224                     return Node.getAttribute(i)
225
226 security.declarePrivate('_getAttributesR')
227 def _getAttributesR(self, Node, styles=[]):
228     style=self._getAttrStyles(Node)
229     if style:
230         styles.append(style)
231     for i in Node.childNodes:
232         self._getAttributesR(i,styles)
233     if styles:
234         return styles

```

Figura 14: Métodos para retirada dos estilos de um elemento Tabela

A linha 215 do trecho de código da Figura 14 faz a declaração de método privado para o método **\_getAttrStyles**, que tem sua assinatura descrita na linha 216, onde recebe como parâmetro um elemento do tipo *Node* do XML. Na linha 220 é efetuada uma condição para verificar se o elemento passado como parâmetro contém atributos. Caso tenha atributos, estes são resgatados e submetidos a uma expressão regular que verifica se o elemento contém estilo. Se existir estilo este é retornado na linha 224.

A linha 227 da Figura 14 mostra a assinatura do método **\_getAttributesR**, que recebe como parâmetro um elemento *Node* e uma lista de estilos. A linha 228 invoca o método **\_getAttrStyles** e atribui seu retorno a variável *style*, que em seguida, se existir, é adicionada na lista *styles*. Esta operação é repetida para todos os elementos filhos do *Node* passado como parâmetro. O retorno deste método consiste é uma lista de estilo.

Após a identificação dos estilos, o trecho de código XML que representa a tabela é retirado em conjunto com seus estilos. Este código é construído novamente em um arquivo ODT em branco, usado como modelo para criação de documentos. Tenta-se resgatar a legenda associada à tabela, retirando trecho de texto localizados acima e abaixo do elemento.

O método **getTableDocumentList** retorna, assim com o método **getImageDocumetList** uma lista de objetos **Grain**.

O *GranulateO3Tool* conta ainda com os métodos *granulateDocument*, *setDocumentMetadata*, *getDocumentMetadata*, *generateHtml*, *generatePdf*, *getSummary*, *getThumbnailsDocument*, *docBuilderO3* e *docBuilderRender*.

O método **granulateDocument** realiza a granularização de um documento, verifica a viabilidade da granularização e invoca os métodos *getImageDocumetList* ou *getPdfImageList* e *getTableDocumentList*.

Os métodos **setDocumentMetadata** e **getDocumentMetadata**, são respectivamente responsáveis pela inserção e o resgate de metadados nos documentos. Eles trabalham em comunicação com o servidor OpenOffice Daemon (*ood*), realizando chamadas aos procedimentos de resgate e inserção de metadados. O procedimento de resgate de metadados tem como retorno um dicionário de metadados suportados pelo *ood*, que são: título, assunto, palavras-chave, descrição, *MIMEType*, autor, referências, versão, idioma. A inserção de metadados suporta os mesmos atributos descritos acima como retorno dos metadados.

Os métodos **generateHtml** e **generatePdf**, realizam respectivamente as operações de

geração de arquivos em formato HTML e PDF. Estes métodos trabalham com recursos disponibilizados pelo ood que suportam a geração de HTML e PDF a partir de arquivos de texto, apresentações e planilhas.

O método **getSummary** faz a captura do sumário de documentos de texto suportado pela ferramenta, exceto PDF. O processo de retirada do sumário consiste na identificação a partir do arquivo *content.xml* da tag *text:h* que define os títulos do documento segundo especificações do padrão ODF. Este método retorna uma lista de tuplas contendo o título e o seu respectivo nível.

O método **getThumbnailsDocument** retira uma imagem em miniatura que representa a primeira página do documento, este método não se aplica a documentos formato PDF.

#### 4.1.3 DocBuilderO3

É uma funcionalidade provida pelo GranulateO3Tool, implementada pelo método **docBuilderO3**, descrito na Figura 15. Este método trabalha na geração automática de documentos formato ODT a partir de referências de objetos e grãos de documentos. Atualmente esta operação suporta somente a geração de documentos a partir de grão tabela.

O método `docBuilderO3` tem comportamento semelhante ao método `getTableDocumentList`, pois o armazenamento do grão tabela é realizado em documento de texto formato ODT, todavia necessita-se extrair a referente tabela do documento ODT para que seja formado um novo documento com todo conteúdo selecionado pelo usuário.

No intervalo entre as linhas 515 e 531 do trecho de código da Figura 15, é mostrado o processo de identificação e retirada de tabela. As tabelas são armazenadas na lista *tables\_self\_service*, que contém um dicionário com a tabela e o estilo associada a ela.

Nas linhas 532, 533 e 534 um documento ODT em branco é criado e atribuído a variável *new\_doc* para receber as tabelas. Este é descompactado para que se acesse o arquivo *content.xml*, e um *parser* é efetuado para identificar o elemento *office-text* que contém o corpo do texto.

Uma interação é realizada entre as linhas 539 e 545 para adicionar as tabelas contidas em *tables\_self\_service* em *new\_doc*, que consiste no retorno do método.

```

510 def docBuilderO3(self,odtfiletables):
511     """
512     Create Document(document as StringIO) generated from list tables
513     """
514     tables_self_service = []
515     for odtfile in odtfiletables:
516         filedoc = StringIO(str(odtfile))
517         odf_file = zipfile.PyZipFile(filedoc,'r')
518         contents = odf_file.read('content.xml')
519         dom = parseString(contents)
520         tables=dom.getElementsByTagName('table:table')
521         stylesDoc=dom.getElementsByTagName('style:style')
522         for t in tables:
523             styles = self._getAttributesR(t)
524             dict_tables_temp = {}
525             styles_temp = []
526             for sty in stylesDoc:
527                 if (sty.getAttribute('style:name') in styles):
528                     styles_temp.append(sty)
529             dict_tables_temp[t]=styles_temp
530             tables_self_service.append(dict_tables_temp)
531         odf_file.close()
532     new_doc = StringIO()
533     template_str=self._createNewOOoDocument()
534     new_doc.write(template_str)
535     template_odt = zipfile.PyZipFile(new_doc,'a')
536     doc = parseString(template_odt.read('content.xml'))
537     office_text=doc.getElementsByTagName('office:text')
538     office_text=office_text[0]
539     for tables_temp in tables_self_service:
540         for tbl,st_list in tables_temp.iteritems():
541             office_text.appendChild(tbl)
542             for st in st_list:
543                 office_automatic_styles=doc.getElementsByTagName('office:automatic-styles')
544                 office_automatic_styles=office_automatic_styles[0]
545                 office_automatic_styles.appendChild(st)
546     template_odt.writestr('content.xml',doc.toxml().encode('utf-8'))
547     template_odt.close()
548     return new_doc

```

Figura 15: Algoritmo de geração automática de documentos - docBuilderO3

#### 4.1.4 DocBuilderRender

O **docBuilderRender** é uma funcionalidade semelhante ao **docBuilderO3**, implementada pelo GranulateO3Tool através do método **docBuilderRender**. Este método faz a geração automática de documentos HTML, a partir de referências de objetos e grãos de documentos, possibilitando a edição via Web de conteúdos utilizando-se um editor integrado ao CMS Plone. A Figura 16 demonstra no editor de HTML Kupu, o resultado da operação de criação *on-the-fly* de documentos HTML.

As referências dos grãos são colocadas no formato ABNT, está é construída a partir dos metadados encontrados nos grãos e por informações do documento fonte do grão.

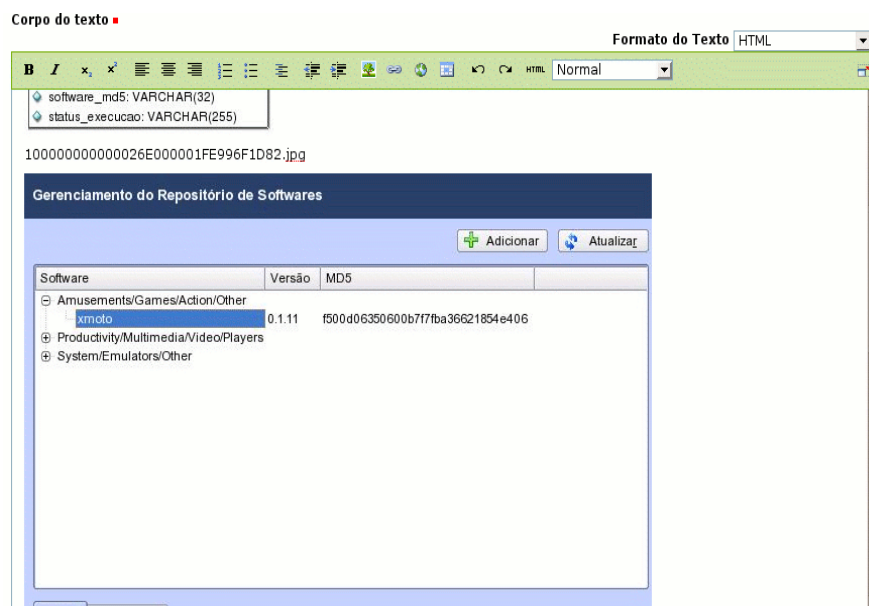


Figura 16: Edição via Web de conteúdos utilizando-se o editor Kupu


## 4.2 GrainCollector

O **GrainCollector** é o processo pelo qual se faz a coleta dos objetos granularizados, essa funcionalidade é implementada com o intuito de apoio a geração automática de conteúdo a partir dos grãos. Tal processo se dá através de uma interface que permite ao usuário selecionar objetos e gerencia-los por meio de uma ferramenta denominada **GrainCart**.

A implementação do **GrainCollector** consiste em dois *scripts* Python denominados **addGrainCollector** e **addGrainCollectorSearch**. Estes *scripts* têm comportamento de *action* para o Plone, sendo o **addGrainCollector** adicionado diretamente na página de exibição dos objetos com características e comportamento de grão de documento, todavia o **addGrainCollectorSearch** é adicionado na página de resultados da busca como demonstrado na Figura 17.

## Search results

Did you not find what you were looking for? Try the

**3 items matching your criteria.** 



-  [Tabela1](#) [1%] by ronaldo, 2007-10-16 12:49 
-  [Table1](#) [1%] by ronaldo, 2007-10-16 12:49 
-  [Table2](#) [1%] by ronaldo, 2007-10-16 12:49 

Figura 17: action resultado de busca - addGrainCollectorSearch

Os scripts **addGrainCollector** e **addGrainCollectorSearch** se baseiam basicamente no uso de *Session* do servidor Zope, onde se armazena as referências dos objetos selecionados através da *action* adicionada aos objetos.

A Figura 18 demonstra o código em Python que implementa o script **addGrainCollector**. Entre as linhas 1 e 7 da Figura 18 é apresentado o cabeçalho do script, onde são definidos títulos e parâmetros.

```
1 ## Script (Python) "addGrainCollector"
2 ##bind container=container
3 ##bind context=context
4 ##bind namespace=
5 ##bind script=script
6 ##bind subpath=traverse_subpath
7 ##title=add item GrainCollector
8
9 r = container.REQUEST
10 session = r.SESSION
11 path = ('/').join(context.getPhysicalPath())
12 objId=path
13
14 grainCollectorList=session.get("grainCollectorList", [])
15
16 if not objId in grainCollectorList:
17     msg = 'portal_status_message=The item has been added with success!'
18     grainCollectorList.append(objId)
19
20 else:
21     msg = 'portal_status_message=Existing item!'
22     r.RESPONSE.redirect('%s?%s' % (context.absolute_url()+'/view', msg))
23
24 session["grainCollectorList"]=grainCollectorList
25 r.RESPONSE.redirect('%s?%s' % (context.absolute_url()+'/view', msg))
```

Figura 18: Script Python addGrainCollector

Na linha 9 o *Request* é capturado para que se possa ter acesso a variável *SESSION* utilizada na linha 10. Nas linhas 11 e 12 uma identificação única do objeto que dispara a *action* é capturada.

Na linha 14 são capturados os valores armazenados na *Session* do *Request*. No intervalo das linhas 16 a 22 uma condição é efetuada para identificar se o referente objeto já existe na *Session*, caso não exista, o objeto é adicionado na lista que representa a *Session* e uma mensagem de confirmação é exibida. Uma vez identificado a existência do objeto, uma mensagem de alerta é enviada ao usuário informando lhe a existência.

### 4.3 GrainCart

É uma interface para gerenciamento dos grãos coletados pelo **GrainCollector**. Este é implementado através de um *portlet* para o Plone e três scripts Python, **removeGrainCollector**, **DBO3** e o **DBEditable**, que permitem ao usuário a exclusão de grãos e a geração de documentos ODT e HTML.

O *portlet* é implementado via ZPT e contém as ações anteriormente citadas. O *script* **DBO3** é responsável pela filtragem e preparação do conteúdo, tendo a ele atribuído a comunicação com o GranulateO3tool, que prover a geração de documentos através do método **docBuilderO3**. O *script* **DBEditable** trabalha na com atribuições semelhantes ao DBO3, sendo ele responsável pela preparação do conteúdo que será submetido ao método **docBuilderRender**, resultando em um HTML, armazenado no diretório do usuário corrente para que este possa ser manipulado.

### 4.4 Instalação

Como pré-requisito para instalação, o GranulateO3Tool necessita que suas dependências estejam instaladas e configuradas, estas são listadas abaixo de acordo com as versões testadas e recomendadas.

- Servidor de Aplicações Zope, versão 2.9;
- Sistema Gerenciador de Conteúdo Plone, versão 2.5.1;
- *Framework* Archetypes, versão 1.4.1-final;
- Produto ATContentType, versão 1.1.3-final;
- Sistema de armazenamento de objetos em sistema de arquivos FileSystemStorage, versão 2.5.6. Este produto é opcional, caso não seja instalado o armazenamento de objetos utilizara o sistema nativo do Zope;
- Servidor OpenOffice Daemon (oood), informações de instalação e configuração são encontradas no *site* <http://www.erp5.org/HowToUseOood>;
- Biblioteca xpdf-common;
- Biblioteca python-extractor;

O procedimento de instalação do GranulateO3Tool consiste primeiramente na obtenção do produto através do repositório SVN `svn://web.cefetcampos.br/nsi/GranulateO3Tool/trunk`, este conteúdo é colocado no diretório de produtos da instância Zope, que por padrão utiliza o diretório *Products*, em um diretório nomeado GranulateO3Tool. Após esta operação necessita-se reiniciar o servidor Zope.

Uma vez efetuado o procedimento acima, a instalação tem continuidade através da interface da instância Plone, onde encontra-se na configuração do portal o **Portal QuickInstaller**.

No **Portal QuickInstaller**, basta acionar a função **Adicionar/Remover Produtos**, selecionar o Produto GranulateO3Tool e pressionar instalar.



## CAPÍTULO V

### ESTUDO DE CASO

Para demonstração do uso da ferramenta criou-se um repositório de objetos de aprendizagem (*learning objects*) para uso no Ensino à Distância (EaD), tal proposta de uso surgir dá necessidade de se criar um ambiente de EaD onde possa disponibilizar objetos de aprendizagem de uma forma que os mesmos atendam um dos propósitos do Ensino à Distância, que é a reusabilidade e a quebra do conteúdo educacional, para serem reutilizados em diferentes ambientes de aprendizagem. O ambiente proposto conta com ferramentas que auxiliam a geração de conteúdo educacional a partir do conteúdo disponibilizado.

#### 5.1 A Solução

Utilizou-se um ambiente baseado na Web, para tal fez-se o uso do servidor de aplicações Zope e o sistema gerenciador de conteúdo Plone. Como dependência da ferramenta GranulateO3Tool, instalou-se um servidor OpenOffice Daemon.

Os servidores foram instalados e configurados em duas máquinas distintas, sendo elas com as seguintes configurações:

##### **Servidor 1**

- Hardware: Intel Pentium 4 1.8Ghz 2 GB RAM;
- Linux Ubuntu 7.04 (kernel 2.6.20-12-server);
- Zope 2.9.6-final;

- Python 2.4.4;
- CMFPlone 2.5.1;
- Archetypes 1.4.1-final;
- ATContentTypes 1.1.3-final;
- FileSystemStorage 2.5.6;
- xpdf-common;
- python-extractor;

## Servidor 2

- Hardware: Intel Pentium 4 2.4Ghz 512MB RAM;
- Mandriva Linux 2007.1 Spring;
- Python 2.5;
- OpenOffice 2.2;
- Xvfb (virtual frame buffer) ;
- OpenOffice Daemon – ood 0.3.0 (svn revision 17429);

Atendendo as necessidade dos objetos de aprendizagem foi desenvolvido um tipo de conteúdo para o Plone, baseado no *framework* Archetypes, nomeado **Document**, demonstrado no diagrama de classes da Figura 19, Este possui como característica principal a utilização das funções de granularização suportada pelo GranulateO3Tool.

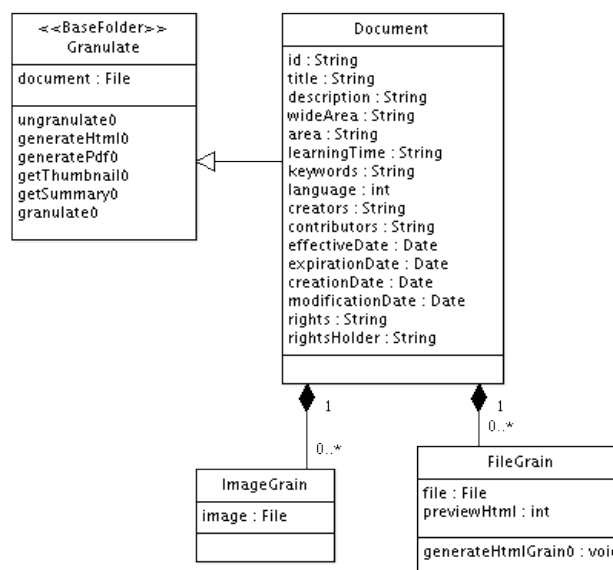


Figura 19: Diagrama de Classe do tipo de conteúdo Document

A classe **Document** reúne alguns metadados do padrão LOM (IEEE, 2002) na definição de seus atributos, sendo estes posteriormente indexados permitindo a busca pelos mesmos. Abaixo são apresentados os índices criados para permitir a recuperação das informações através da máquina de busca padrão do Plone.

- Título;
- Palavras-chave;
- Objetivo;
- Autor(s);
- Idioma;
- Direitos Autorais;
- Propriedade Industrial;
- Grande Área de Conhecimento
- Área de Conhecimento;

A classe **Granulate**, apresentada no diagrama da Figura 19, prove todo acesso a ferramenta GranulateO3Tool, tendo a ela atribuída a responsabilidade pelo processo de granularização e pelo acesso a outras funcionalidades do GranulateO3Tool.

As classes **ImageGrain** e **FileGrain**, são responsáveis pelo armazenamento e tratamento dos grãos retirados dos documentos, estas possuem os metadados herdados ou não do documento pai.

### 5.1.1 Armazenamento

O armazenamento dos objetos e índices de catálogo do Zope é feito na sua base de dados de objetos, ZODB, todavia a estratégia de persistência de dados em um único arquivo(FileStorage), adotada pelo ZODB, torna o armazenamento de grande quantidade de objetos desconfortável no que se refere a gerenciamento de documentos.

Com objetivo de flexibilizar o gerenciamento, adotou-se o uso de um componente do Plone denominado *FileSystemStorage* . Este componente usa a filosofia de armazenamento em sistema de arquivos, garantindo acesso de leitura e escrita ao sistema de arquivos, contudo apenas para o armazenamento do documento em si no sistema de arquivos, entretanto dados relacionados ao documento como metadados são armazenados na camada do banco de

objetos. Tal escolha se justifica no argumento de CARVALHO (2007) onde afirma que nenhum software é melhor do que o sistema operacional para armazenar e servir uma grande coleção de arquivos, tornando-o uma ótima escolha, demonstrando que um bom método de recuperação de informação é oferecido.

### 5.1.2 Processo de Granularização

Um *workflow* foi desenvolvido para o tratamento e automação do fluxo de trabalho dos objetos de aprendizagem. Este é representado pelo diagrama de estado da Figura 20, onde se demonstra todos os processos envolvidos, desde o envio de conteúdo, sua granularização e publicação.

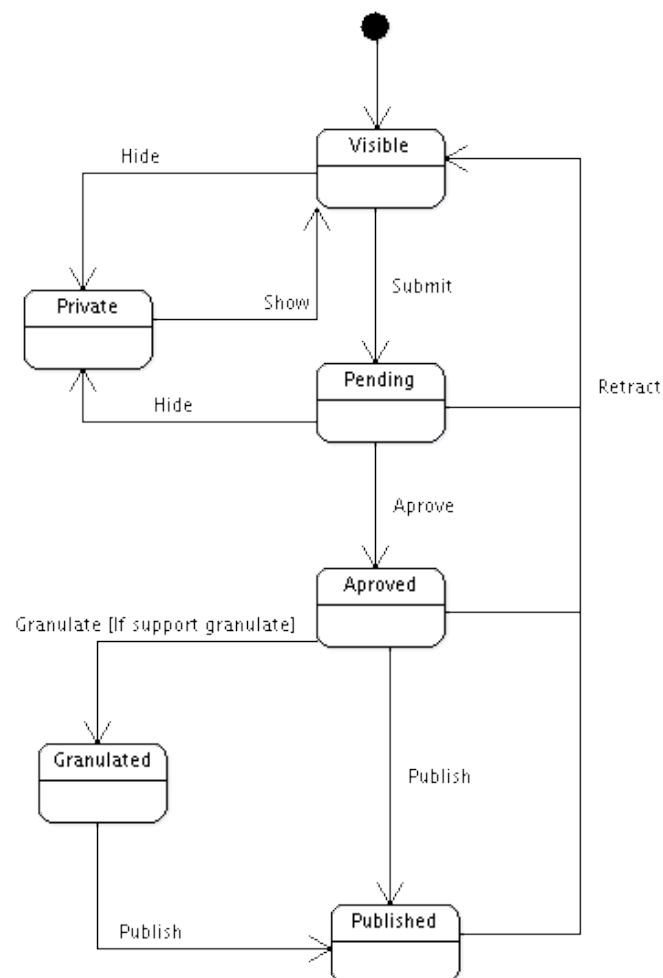


Figura 20: Processo de Granularização de Documentos

O documento ao ser inserido no portal tem seu estado inicial *Visible*, onde somente o dono do documento, o revisor e os administradores do portal têm acesso a essa informação. Este pode ser submetido a avaliação de um revisor ou tornar-se privado, passando assim para o estado *Private*. Ao submeter o documento, este entra no estado de *Pending*, onde um usuário com papel de revisor pode aprovar o documento, passando-o para o estado *Aproved*, ou retrain, passando este para o estado inicial *Visible*.

No estado *Aproved* uma transição é disparada e se inicia o processo de granularização do documento. Após o processo de granularização, o documento passa para o estado *Granulated*, em seguida este efetua uma transição automática para o estado *Published*, onde seu conteúdo é disponibilizado no portal. A transição para o estado *Granulated* ocorre somente para documentos suportados pela ferramenta GranulateO3Tool.

## 5.2 Utilização

Para inserir conteúdo no portal é necessário que o usuário esteja autenticado, pois não ocorrendo a identificação automática do autor, a ferramenta sugere o usuário autenticado como autor.

Uma das característica da solução, citada no Capítulo IV, é o suporte a diversos tipos de arquivos, entretanto os recursos de granularização de imagem e tabelas somente serão realizados com documentos de texto, sendo este documento um arquivo no formato PDF, somente os grãos imagem serão identificados e retirados. Na Figura 21 é demonstrado a tela de inserção de conteúdo.



Figura 21: Tela de Inserção de Arquivo

Após carregar o arquivo para o Portal e salvar, a ferramenta verifica o tipo de arquivo inserido e sendo este um arquivo no formato ODF, uma comunicação é estabelecida com o servidor OpenOffice.org Daemon para obter os metadados embutidos no documento. Caso o documento seja formato PDF, os metadados são obtidos sem a necessidade de comunicação com o servidor OpenOffice Daemon.

Concluída a obtenção dos metadados o usuário é automaticamente direcionado para uma tela de edição dos mesmos, onde são exibidos os dados obtidos na operação anteriormente citada. Ver Figura 22.

Contents View Edit Properties

Actions Add item State: Public Draft

Changes saved.

### Edit Documento

This information, also referred to as metadata is the collection of information that is used to categorize an object, assign effective dates and expiration dates, language, and keywords.

**Title**

Ontology Engineering

**Allow Discussion on this Item**

☒ Default  
☐ Enabled  
☐ Disabled

**Keywords**

Existing keywords

- Informação em Saúde do Trabalhador
- Informática em Saúde Pública
- Levantamentos Epidemiológicos
- Ontologia**
- Ontologia Web Semântica**
- Organização e Administração
- Serviços de Informação
- Sistemas de Apoio à Decisão
- Software
- Vigilância da População
- ciclo de vida
- customização
- decisão
- implementação

New keywords

**Objetivo**

A short summary of the content

Neste documento, o conceito de ontologia é descrito e estudamos uma abordagem para a construção de Ontologias. A abordagem seguida é a descrita em "Ontology Development 101: A Guide to Creating Your First Ontology" by Natalya F. Noy and Deborah L. McGuinness. Também são abordadas as diferentes linguagens disponíveis e algumas ferramentas de

Figura 22: Tela de Edição de Metadados

Efetuada o processo de edição, o documento entra no estado de esboço público, esse processo se dá através da utilização do *workflow* apresentado anteriormente.

Para que o documento possa se granularizado, necessita-se que ocorra a mudança de seu estado para aprovado. Pode-se atribuir papéis distintos na utilização do Portal, tendo em vista que o processo de granularização é disparado por um revisor que aprova o documento, ocasionado assim a sua mudança de estado.

Encontrando-se no estado de aprovado, o processo de granularização do documento é disparado e seu conteúdo é publicado no Portal com seus respectivos grãos. A Figura 23 demonstra a tela de exibição do documento.

## Ontology Engineering

by nsi — last modified 2007-09-19 11:06 [History](#)

**Objetivo:**  
Neste documento, o conceito de ontologia é descrito e estudamos uma abordagem para a construção de Ontologias. A abordagem seguida é a descrita em "Ontology Development 101: A Guide to Creating Your First Ontology" by Natalya F. Noy and Deborah L. McGuinness. Também são abordadas as diferentes linguagens disponíveis e algumas ferramentas de trabalho, para a construção de Ontologias.

**Documento:**  
 Monografia.odt (OpenDocument Text 192Kb)

**Propriedade Industrial:**

**Grande Area:**  
Linguística, Letras e Artes

**Area:**

**Tempo de Aprendizagem:**

**Modo De Entrega:**

**Finalidade:**

10000000000017100000EE9DE4C195.png — by nsi — last modified 2007-09-19 11:06  
 10000000000011200000192FF8DFD73.png — by nsi — last modified 2007-09-19 11:06  
 1000000000002C10000073D3F53A96.png — by nsi — last modified 2007-09-19 11:06  
 1000000000001B900000F6B0B284A4.png — by nsi — last modified 2007-09-19 11:06  
 1000000000001B500000ED7EF72FDA.png — by nsi — last modified 2007-09-19 11:06

*Figura 23: Tela de exibição de documentos*

Os grãos Tabela e Imagem têm uma interface própria de exibição, onde se obtêm uma pré-visualização do conteúdo, como demonstrado nas Figuras 24 e 25.

Para se obter a pré-visualização da tabela, um processo de conversão de arquivos é efetuado, gerando o código HTML que representa a tabela, entretanto, ocorre que durante este processo, algumas informações relativas a formatação da tabela são perdidas.

### Table1

by [ronaldo](#) — last modified 2007-10-30 15:53

#### Objetivo:

#### Propriedade Industrial:

#### Grande Area:

#### Area:

#### Tempo de Aprendizagem:

#### Modo De Entrega:

#### Finalidade:

#### File:

[Table1 \(OpenDocument Text 19Kb\)](#)

#### Document Preview:

Descrição	Data
Busca de inovações no processo produtivo desenvolvidas por instituições de pesquisas.	Segundo semestre de 2008
Incorporação de melhorias no processo produtivo	Primeiro semestre de 2010
Obter níveis de qualidade no produto final afim de alcançar certificações.	Segundo semestre de 2008
Eliminar a informalidade da cadeia produtiva	Primeiro semestre de 2009
Reduzir emissão de poluentes	Primeiro semestre de 2010
Elaboração de políticas voltadas para a preservação da vegetação nativa da região	Segundo semestre de 2008
Investimento em marketing na região norte e noroeste	Primeiro semestre de 2009

Figura 24: Tela de visualização do grão Tabela

A interface do grão imagem assim como o grão tabela, possui uma pré-visualização do conteúdo, gerada através funções específicas do Plone para tratamento de imagem, e contém os metadados herdados ou não do documento de origem.

### UbuntuLogo.png

by [ronaldo](#) — last modified 2007-11-03 17:44

#### Objetivo:

#### Propriedade Industrial:

#### Grande Area:

Linguística, Letras e Artes

#### Area:

Artes

#### Tempo de Aprendizagem:

#### Modo De Entrega:

#### Finalidade:

#### Image:



[Current image](#) PNG image — 33 KB

Figura 25: Tela de visualização do grão Imagem



### 5.3 Geração de Conteúdo

A geração de conteúdo se realiza através de quatro processos citados no Capítulo IV: o GrainCollector é o processo pelo qual se efetua a coleta dos grãos, o GrainCart faz o gerenciamento dos grãos, o docBuilderO3 gera o documento em formato ODT e o docBuilderRender gera um documento em formato HTML que pode ser manipulado pelo editor integrado ao Plone, o Kupu.

Isso torna a ferramenta de grande ajuda na elaboração de conteúdo educacional como mostrado adiante.

Todo grão e documento possui uma ação que é disparada através de um botão em forma de “carrinho de compras”, localizado na parte superior da visualização do conteúdo e no resultado da busca.

Ao ser acionado uma referência do documento é adicionada ao GrainCart, como pode ser vista na Figura 26.

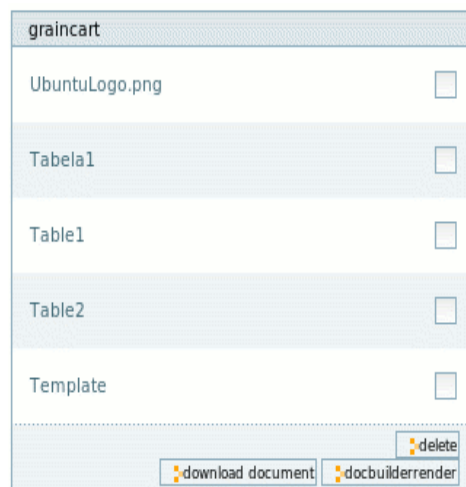
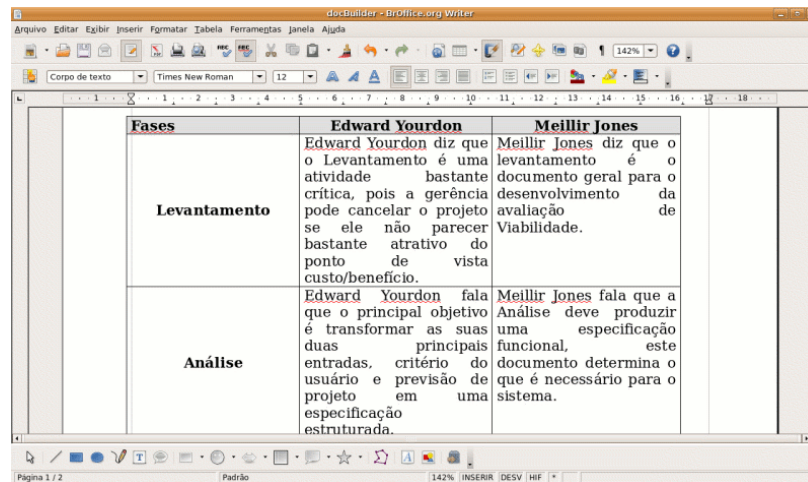


Figura 26: Portlet GrainCart

O gerenciamento é efetuado no *portlet* do GrainCart (Figura 26), onde se pode realizar a exclusão de grãos e documentos, e optar pela forma de geração de conteúdo a partir dos grãos, sendo estas o docBuilderO3 e docBuilderRender.

Para gerar o documento com o conteúdo referenciado no GrainCart pelo método docBuilderO3, basta acionar o botão Download Document, este dispara o processo de construção do documento passando uma lista dos objetos referenciados no GrainCart para o

método docBuilderO3, onde o conteúdo é capturado para que se possa construir um novo documento. O resultado de tal operação pode ser vista na Figura 27, onde se demonstra no editor de textos OpenOffice.org o documentos gerado pelo docBuilderO3.



Fases	Edward Yourdon	Meillir Jones
Levantamento	Edward Yourdon diz que o Levantamento é uma atividade bastante critica, pois a gerência pode cancelar o projeto se ele não parecer bastante atrativo do ponto de vista custo/beneficio.	Meillir Jones diz que o levantamento é o documento geral para o desenvolvimento da avaliação de Viabilidade.
Análise	Edward Yourdon fala que o principal objetivo é transformar as suas duas principais entradas, critério do usuário e previsão de projeto em uma especificação estruturada.	Meillir Jones fala que a Análise deve produzir uma especificação funcional, este documento determina o que é necessário para o sistema.

*Figura 27: Tela OpenOffice.org, tabela gerada a partir do docBuilderO3*

O método de construção docBuilderRender trabalha de forma semelhante ao docBuilderO3, entretanto o resultante da operação é um arquivo HTML, que pode ser editado através do editor Kupu integrado ao Plone.

Ao se acionar o botão docBuilderRender, demonstrado na Figura 26, um HTML é gerado *on-the-fly* com o conteúdo do GrainCart tendo suas referências inseridas em formato ABNT. Automaticamente este conteúdo é salvo em uma área específica do usuário autenticado, podendo ser editado através do Kupu, onde se possibilita a adição de novos grãos imagem e tabela.

## CAPÍTULO VI

### CONCLUSÃO

Este trabalho demonstrou o desenvolvimento de uma ferramenta de caráter inovador, possibilitada através de tecnologias *Open Source*, e que faz parte de uma solução de ECM desenvolvida no Núcleo de Pesquisas em Sistemas de Informação – NSI, CEFET Campos , onde se agrega o uso de recursos computacionais na automatização de procedimentos antes efetuados de forma manual, todavia sua aplicabilidade e demonstrada no estudo de caso, onde um repositório de objetos de aprendizagem é criado para o uso no Ensino à Distância.

Este estudo apresenta também uma ferramenta de apoio a geração de conhecimento e aborda um dos pilares do Ensino à Distância que é a reusabilidade do conteúdo como um todo e em partes.

O GranulateO3Tool se apresenta como uma ferramenta base para os projetos Biblioteca Digital da EPT, Centro de Documentação Digital da EPT e Observatório Nacional da EPT, desenvolvidos no CEFET Campos.

#### 6.1 Estudos Futuros

Propõem-se como estudos futuros o incremento de novas funcionalidade ao GranulateO3Tool e o melhoramento de sua interface com o usuário. Destaca-se a necessidade de efetuar a retirada de tabelas em documentos PDF, pois este formato representa uma grande parcela da informação hoje gerada. Entretanto surge a necessidade de torna a solução escalável, se tratando da manipulação de arquivos PDF.

Sugere-se a criação de *templates* e macros no formato OpenDocument para facilitar o processo de estruturação da informação e viabilizar a retirada de capítulo, parágrafos e referências bibliográficas, auxiliando o usuário na construção de seus documentos.

Transpor o GranulateO3Tool para serviço de chamada de procedimento remoto, implementado através do protocolo XML-RPC, para que se alcance a interoperabilidade entre sistemas, levando o seu uso em diferentes ambientes.

Aprimorar o processo de coleta e gerenciamento de grãos, providos pelo GrainCollector e o GrainCart, tornando estes processos mais rápidos, interativos e amigáveis ao usuário.

## REFERÊNCIAS

BORGES, Alessandro dos Santos; 2006 <http://pt.wikipedia.org/wiki/Dados> acessado em 18 de setembro de 2007

BECK, R.J. Learning Objects: What?. Center for International Education. University of Wisconsin. Milwaukee. 2001

BURTON, Joel; 2007 <http://plone.org/about/plone/> acessado em 15 de setembro de 2007

CARVALHO, Rogério Atem de; 2007. An Enterprise Content Management Solution Based on Open Source. Research and Practical Issues of Enterprise Information System II, Volume 1, Edited by Li Xu, A Min Tjoa, Sohail Chaudhry

COSTI, Ana Maria Mallman, et al. Valoração de Soluções em Tecnologia da Informação com Base no Conceito de Capital Intelectual. In: SANTOS, Antônio Raimundo dos, et al. Gestão do conhecimento: uma experiência para o sucesso empresarial, Curitiba, 2001, p. 129-167.

FERRI, Jean Rodrigo. Site para o Projeto Crase. Ijuí, 2001. Trabalho de Conclusão de Curso I (Bacharelado em Informática) - DETEC, Unijuí - RS.

FERRI, Jean Rodrigo. Ambiente para Construção de textos (ACT) : Um Ambiente para a Construção Colaborativa e Publicação de Textos na Web . Ijuí, 2002. Trabalho de Conclusão de Curso (Bacharelado em Informática) - DETEC, Unijuí - RS.

IEEE, 2002. [http://ltsc.ieee.org/wg12/files/LOM\\_1484\\_12\\_1\\_v1\\_Final\\_Draft.pdf](http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf) acessado em 10 de setembro de 2007

MINETTO, Elton Luíz; 2004; Automação de tarefas do OpenOffice usando o Python; <http://www.linhadecodigo.com.br/ArtigoImpressao.aspx?id=388> acessado em 15 de setembro de 2007

MORAN, José Manuel. Textos sobre O que é educação a distância em <http://www.eca.usp.br/prof/moran/dist.htm> acessado em 18 de setembro de 2007.

MORAN, José Manuel. Textos sobre Educação inovadora presencial e a distância em [http://www.eca.usp.br/prof/moran/inov\\_1.htm](http://www.eca.usp.br/prof/moran/inov_1.htm) acessado em 18 de setembro de 2007.

MUZIO, J.; HEINS, T.; MUNDELL, R. Experiences with Reusable eLearning Objects: From Theory to Practice. Victoria, Canadá. 2001.

MONNERAT, Rafael Manhães, LEAL, William de Matos. Integração de Ferramentas de Geração Automatizada de Portais Web e de Gerenciamento de Informação Pessoal . Campos dos Goytacazes, 2004. Trabalho de Conclusão de Curso (Graduação em Informática)-CEFET-Campos - RJ.

OLIVEIRA , Lia Raquel; BLANCO, Elias; 2003. A Propósito de Elearning e de Campus Virtual; <http://www.nonio.uminho.pt/actchal03/05comunicacoes/Tema2/09LiaOliveira.pdf> acessado em 18 de setembro de 2007.

RIBEIRO, Ticianne de Gois, BRAGA, Antonio Rafael, PIMENTEL, Veronica Lima; 2006 Um Ambiente Virtual de Ensino e Aprendizagem Baseado em Web Semântica e Web Services; [http://www.artigocientifico.com.br/uploads/artc\\_1160063078\\_11.pdf](http://www.artigocientifico.com.br/uploads/artc_1160063078_11.pdf) acessado em 15 de julho de 2007.

ROBREDO, Jaime. *Working Knowledge*, Boston, 11a. ed. Harvard Business School Press, 1998.

SALIM, Jean Jacques; SABBAG, Paulo Yazigi (2007) <http://www.fgvsp.br/conhecimento/>

[home.htm](#) acessado em 18 de setembro de 2007.

SILVA, Renata Mesquita da. Um Método para Modelagem de Sistemas de Suporte à Decisão. Campos dos Goytacazes, 2005. Curso de Pós-Graduação Lato-sensu em Produção e Sistemas - CEFET-Campos - RJ.

SETZER, Valdemar W (1999). Dado, Informação, Conhecimento e Competência, [http://www.dgz.org.br/dez99/Art\\_01.htm](http://www.dgz.org.br/dez99/Art_01.htm) acessado em 18 de setembro de 2007

SVARRE, Klaus 2006. [http://searchwebservices.techtarget.com/sDefinition/0,,sid26\\_gci508916,00.html](http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci508916,00.html) acessado em 15 de setembro de 2007